Frank Juergen Knaesel

Um Modelo para Compartilhamento Consistente de Dados em Ambientes de Computação Móvel

Florianópolis - SC

Julho de 2007

UNIVERSIDADE FEDERAL DE SANTA CATARINA PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Frank Juergen Knaesel

Um Modelo para Compartilhamento Consistente de Dados em Ambientes de Computação Móvel

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Mário Antônio Ribeiro Dantas, Dr.

Florianópolis, Julho de 2007.

Um Modelo para Compartilhamento Consistente de Dados em Ambientes de Computação Móvel

Frank Juergen Knaesel

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Raul Sidnei Wazlawick, Dr.

Coordenador do Curso

Banca Examinadora

Prof. Mário Antônio Ribeiro Dantas, Dr. (Orientador)

Prof. Carlos Barros Montez, Dr.

Prof. Frank Augusto Siqueira, Dr.

Prof. João Bosco Mangueira Sobral, Dr.

Agradecimentos

Gostaria de agradecer primeiramente a Deus, por estar comigo em todos os momentos, sobretudo nos mais difíceis.

À minha esposa Angela e ao meu filho Miguel, por terem me mostrado o verdadeiro sentido do amor e da vida, e pelo incentivo incondicional na caminhada e na conclusão de mais esta importante etapa de minha vida.

Aos meus pais Norberto e Ingelore, pela educação e pelos exemplos de honestidade que norteiam minha caminhada.

Ao meu orientador, Prof. Mário Dantas, pela sua orientação, dedicação e disponibilidade, pela credibilidade depositada em mim, pelas conversas incentivadoras ao longo do trabalho realizado, por seus ensinamentos e também pelo seu bom humor constante.

Aos colegas do LaPeSD: Vinícius, Parra, Anúbis e Jeferson, pelos momentos de descontração, companheirismo, conselhos, sugestões e momentos produtivos no LaPeSD.

À Vera Sodré (Verinha) pela constante simpatia e prestatividade em todos os momentos.

À FAMEG (Faculdade Metropolitana de Guaramirim), pelo apoio e abertura do espaço para que este trabalho fosse realizado.

E a todos aqueles que contribuíram, direta ou indiretamente, para que este trabalho fosse realizado, meus sinceros agradecimentos.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

WAP - *Wireless Application Protocol*

IP - *Internet Protocol*

PDA - Personal Digital Assistants

SO - Sistema Operacional

NS2 - *Network Simulator*

RTT - *Round Trip Time*

P2P - *peer-to-peer*

IM - *Instant Messenger*

VoIP - Voz sobre IP (*Internet Protocol*)

IP - *Internet Protocol*

WLANs - *Wireless Local Area Network*

AMPS - *Advanced Mobile Phone Systems*

TACS - *Total Access Communication System*

NMT - *Nordic Mobile Telephone*

GSM - *Global System for Mobile Communication*

TDMA - *Time Division Multiple Access*

CDMA - *Code Division Multiple Access*

ETSI - *European Telecommunications Standard Institute*

UMTS - *Universal Mobile Telecommunication System*

WAP - *Wireless Application Protocol*

WML - *Wireless Markup Language*

HTML - *Hyper Text Markup Language*

WMLScript - *Wireless Markup Language Script*

WTA - *Wireless Telephony Application*

WTAI - *Wireless Telephony Application Interface*

WAE - *Wireless Application Environment*

WWW - *World Wide Web*

WSP - *Wireless Session Protocol*

WTP - *Wireless Transaction Protocol*

WTLS - *Wireless Transport Layer Security*

TLS - *Transport Layer Security*

SSL - *Secure Socket Layer*

WDP - *Wireless Datagram Protocol*

Wi-Fi - *Wireless Fidelity*

MAC - *Media Access Control*

LAN - *Local Area Network*

IBSS Networks - *Independent Basic Service Set*

ESS Networks - *Extended Service Set*

Mbps - *Megabits per second*

ACK - Acknowledge

MANETs - *Mobile Ad-Hoc Networks*

SMH - *Small Mobile Host*

LMH - *Large Mobile Host*

SGBD - Sistemas Gerenciadores de Banco de Dados

QoS - *Quality of Service*

NTP - *Network Time Protocol*

RID - *Record IDentificator*

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

WARD - *Wide Area Replication Domain*

CFS - *Coda File System*

IPP - *Interleaved Puss and Pull*

NTP - *Network Time Protocol*

XML - eXtensible Markup Language

RPCC - *Relay Peer based Cache Consistency*

FAMEG - Faculdade Metropolitana de Guaramirim

ERP - *Enterprise Resource Planning*

SIGAF - Sistema Integrado de Gestão Acadêmica e Financeira

PHP - Hypertext Pre-Processor

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

UML - *Unified Modeling Language*

LDB - Lei de Diretrizes e Bases da Educação

Resumo

Em ambientes de computação móveis, assistentes pessoais digitais podem gravar dados localmente para possibilitar o trabalho do dispositivo móvel mesmo quando ele não está na área de abrangência de uma rede sem fio. Estes mesmos dados gravados localmente podem ser compartilhados entre dois ou mais usuários, sendo necessários mecanismos para prover consistência neste compartilhamento de dados. Assim, é importante ter sempre à mão os dados mais recentes.

Para resolver este problema, foi desenvolvido um mecanismo para atualização de dados em dispositivos móveis a partir de um servidor estacionário, usando um protocolo de *timestamp*. Tal mecanismo foi testado em um ambiente experimental, que possui um servidor em uma rede estruturada e dois clientes móveis buscando e enviando dados entre o cliente móvel e o servidor (quando este estiver na área de abrangência da rede sem fio estruturada) e entre os clientes móveis (quando não há cobertura pela rede sem fio estruturada) usando uma rede ad-hoc.

Nesta dissertação, o modelo de compartilhamento de dados entre cliente e servidor, e entre os próprios clientes é descrito em detalhes, bem como o ambiente experimental e o desenvolvimento do protótipo.

Palavras-chave: Computação Móvel, Compartilhamento de Dados, Consistência, Redes *Wireless*, Redes *Ad-Hoc*.

Abstract

In mobile wireless environments, personal digital assistants may cache data to work even when the mobile client is far from the wireless network covering area. The same cached data locally stored may be shared by two or more users, and it is necessary to have some mechanisms to provide consistency in this sharing of data. So, it is important to have always the most updated data.

To solve this problem, we developed a mechanism for data sharing between the server and mobile devices using a timestamp protocol to have always the most updated data. The experimental environment has one server in a structured network and two mobile clients getting and sending data between client and server (when it is covered by the structured wireless network) and between the mobile clients (when a structured wireless network is absent) using an ad-hoc network.

In this dissertation, it is described in details the mechanism used to share data between client and server, and between the clients itself, as well as the experimental environment and prototype software.

Keywords: Mobile Computing, Data Sharing, Consistency, Wireless Ad-Hoc Networks.

1 Introdução

Aplicações distribuídas que fazem uso de dados também distribuídos são atualmente encontradas nas mais variadas áreas do conhecimento. Esta tendência está relacionada a questões como o avanço tecnológico, economia de recursos e otimização do tempo. Questões como alta disponibilidade e desempenho têm provocado um aumento no uso destes sistemas.

Nos últimos anos, também tem-se verificado um aumento expressivo na comercialização de dispositivos computacionais móveis. Isto se deve à principal característica destes equipamentos, que é sua mobilidade. As aplicações existentes para estes dispositivos são as mais variadas: desde simples agendas e blocos de notas a aplicações que fazem uso de uma rede sem fio acessando grandes bases de dados.

Segundo Itani (ITANI; DIAB; ARTAIL, 2005b) aplicações móveis são executadas em dispositivos móveis tais como PDA (Personal Digital Assistants), telefones inteligentes (*smart phones*) e telefones celulares comuns. Existem também três tipos de aplicações que podem ser executadas nestes dispositivos:

- aplicações que são executadas sem comunicação com outros dispositivos;
- aplicações que são executadas com comunicação intermitente através de uma rede cabeada ou sem fio;
- aplicações que são executadas com conexão permanente através de uma rede sem fio.

Esta mobilidade implica em um novo paradigma de desenvolvimento de software, devido à limitação de recursos destes dispositivos, quando comparados a computadores *desktops* ou

notebooks, bem como à adoção de novos mecanismos para gerenciamento dos dados. Estes mecanismos envolvem a criação e a manutenção de réplicas de dados que devem ser o mais fiéis possíveis quando comparados aos sistemas cliente/servidor tradicionais, uma vez que normalmente os dados utilizados nestes dispositivos móveis provém de sistemas cliente/servidor tradicionais.

O principal componente deste tipo de aplicação móvel é o armazenamento persistente de dados. Esta tecnologia permite que o cliente móvel possa trabalhar com dados previamente armazenados no dispositivo que foram buscados através de uma conexão cabeada ou sem fio, sem a necessidade constante de estarem permanentemente conectados à rede estruturada. Segundo (ITANI; DIAB; ARTAIL, 2005b), para realizar este armazenamento persistente, é possível usar o sistema de arquivos do próprio dispositivo, adquirir uma solução comercial de banco de dados para dispositivos móveis ou implementar sua própria solução.

Assim, segundo (ITANI; DIAB; ARTAIL, 2005a), em um ambiente de computação móvel sem fio, os clientes móveis podem usar os dados localmente armazenados para trabalhar mesmo quando não houver uma rede estruturada sem fio disponível. Isto pode ser feito através de um processo de replicação de dados, criando diversas cópias locais dos dados que são chamadas de réplicas, ou como alguns autores (GUPTA; SRIMANI, 2001; PENG; CHEN, 2005; SONG; PARK; YANG, 2006) preferem, também podem ser chamadas de *cache*.

Normalmente, o processo de replicação é inicializado a partir de um ou mais servidores para os clientes, e o processo de reconciliação, no qual um cliente envia de volta ao servidor os dados que foram alterados pela aplicação no cliente móvel, é iniciado a partir do próprio cliente para o servidor. Depois de concluído o processo de reconciliação, o(s) servidor(es) precisam re-atualizar os dados recebidos para os outros clientes. As razões para este processo acontecer desta forma são:

- reduzir o uso constante da rede e conseqüentemente economizar tempo de bateria;
- evitar desconexões freqüentes que podem ocorrer devido à instabilidade do sinal da rede sem fio.

Alguns pesquisadores (HOU et al., 2001; YUEN et al., 2000; XU; TANG; LEE, 2003) afirmam que o armazenamento local de dados é um método interessante para reduzir a quantidade de dados no processo de comunicação, assim como a latência no acesso aos dados e a largura de banda limitada disponível para os dispositivos móveis. Este armazenamento local reduz o tempo de acesso aos dados, caso os dados necessários estejam armazenados localmente.

Entretanto, a estabilidade da rede e a grande largura de banda necessária em mecanismos já existentes de gerenciamento de cache em aplicações que fazem uso de bancos de dados distribuídos entram em conflito com a mobilidade provida pelos dispositivos móveis, com a limitada largura de banda disponível para estes dispositivos e com a instabilidade de sinal das redes sem fio. Assim, o desenvolvimento de estratégias para o compartilhamento consistente de dados torna-se um desafio (XU; TANG; LEE, 2003).

Como mencionado em (BERKENBROCK; DANTAS, 2005; CUNHA; DANTAS, 2004), o aumento do número de usuários buscando e modificando os dados, força o planejamento cuidadoso de sistemas de gerenciamento de banco de dados distribuídos. Isto deve ocorrer para garantir altos níveis de confiabilidade e disponibilidade no processo de atualização dos dados.

Métodos para pesquisa e atualização de dados em ambientes móveis diferem da arquitetura tradicional cliente/servidor (LAUZAC; CHRYSANTHIS, 2002). Este fato ocorre principalmente por causa da alta mobilidade em redes sem fio, pelos requisitos dos usuários em termos de exatidão nos dados e o montante que o usuário deseja investir em infraestrutura de comunicação sem fio.

1.1 Objetivos

Pelos motivos expostos, esta dissertação tem por objetivo propor um mecanismo para prover consistência e confiabilidade no compartilhamento de dados entre clientes móveis e servidor, bem como entre os próprios clientes móveis e que seja da mesma forma bastante leve, utilizando ao mínimo a largura de banda disponível na rede sem fio, visto que esta é extremamente limitada quando comparada a sistemas de rede cabeados. A idéia central é que um dispositivo móvel possa atuar como servidor para outros dispositivos móveis. A comunicação entre o servidor e

os dispositivos móveis poderá ser feita através de uma rede sem fio estruturada e também em redes ad-hoc, caso não existam pontos de acesso sem fio disponíveis na área de cobertura do dispositivo móvel.

Como contribuição secundária, foram desenvolvidas interfaces gráficas aos usuários dos dispositivos móveis para realizar a consulta e atualização dos dados no cache local de forma amigável.

1.2 Estrutura do Documento

A estrutura desta dissertação está dividida em duas partes. A primeira parte apresenta uma revisão bibliográfica abordando os principais tópicos relacionados a computação móvel, redes sem fio e a coerência e consistência de cache nos dispositivos móveis, bem como o mecanismo de *timestamp* que será utilizado para prover esta consistência.

A segunda parte apresenta alguns trabalhos correlatos sobre o assunto e descreve o modelo proposto para este compartilhamento consistente de dados baseado em um mecanismo de *timestamp* que será utilizado entre cliente e servidor e também entre os próprios clientes. Nesta parte também é descrito o ambiente experimental e apresentado o protótipo do software que foi utilizado para verificação prática do modelo ora apresentado, e a avaliação deste sistema através de várias consultas e atualizações submetidas a estes dispositivos.

Estas duas partes estão organizadas nos capítulos conforme descritos a seguir:

- O capítulo 2 está dividido em três seções:
  - a primeira seção apresenta as principais características de ambientes de computação móvel;
  - a segunda seção abordados os modelos de comunicação utilizados nestes ambientes;
  - na terceira seção, são apresentadas informações sobre redes sem fio e seus modos de operação.

- O capítulo 3 trata sobre aspectos de confiabilidade dos dados e contém as seguintes seções:
  - coerência de *cache*;
  - consistência;
  - replicação de dados, seus modelos e protocolos;
  - particionamento;
  - disseminação;
  - reconciliação;
  - mecanismo de *timestamp*.
- No capítulo 4, o modelo proposto é apresentado em detalhes. Neste capítulo são apresentados os trabalhos correlatos, bem como descritas as principais características do modelo, a arquitetura desenvolvida e o protótipo implementado. Neste capítulo também será descrito em detalhes o ambiente experimental utilizado para demonstrar o comportamento do sistema proposto através de testes empíricos;
- Finalmente, no capítulo 5 são apresentadas as conclusões, as limitações e problemas encontrados, bem como propostas de continuação deste trabalho.

2 *Computação Móvel e Redes Sem Fio*

2.1 Computação Móvel e suas Características

A computação móvel, também conhecida como computação nômade, um novo paradigma de computação distribuída advinda das tecnologias de redes sem fio e aumento do poder de processamento e capacidade de armazenamento dos dispositivos móveis, já é uma realidade em vários segmentos educacionais, empresariais e governamentais. Isto possibilitou a utilização que alguns autores (TRIFONOVA; RONCHETTI, 2006; PATTEN; PASSERINI, 2005; AGRAWAL; SREENAN, 1999) denominam de *anytime-anywhere-computing*. Desde que seja possível estabelecer uma comunicação com a rede, seja ela estruturada ou ad-hoc, um dispositivo móvel pode receber ou enviar dados a qualquer tempo e qualquer local.

De acordo com (SATYANARAYANAN, 2001), podem ser aplicados à computação móvel, inúmeros princípios básicos da computação distribuída, tais como: comunicação remota, tolerância a falhas, alta disponibilidade, acesso remoto a informações e segurança. Entretanto, a mobilidade também gera algumas restrições devido a características destes dispositivos e fazem com que novas estratégias sejam adotadas para resolver estes problemas. Conforme (BARBARA, 1999), dentre estas limitações podem ser citadas: a variação da qualidade do sinal de rádio em redes sem fio, menor capacidade de processamento e armazenamento quando comparados a computadores *desktop*, e o fato de possuírem uma fonte de energia com duração curta.

Apesar disto, estas limitações servem de base para que novos desafios sejam enfrentados no sentido de minimizar os efeitos negativos destas limitações e podem se transformar em questões interessantes para projetos de gerenciamento de bancos de dados móveis distribuídos. Em ou-

Figura 1: Componentes Fundamentais da Computação Móvel

tras palavras, estas características causam mudanças significativas de infraestrutura na qual tais sistemas são construídos. Como escrito por (OZSU; VALDURIEZ, 1999), estas mudanças estão relacionadas ao gerenciamento de diretórios, difusão de dados, processamento e otimização de consultas, gerenciamento de transações, dentre outros.

Segundo (LIU; MARLEVI; MAGUIRE, 1995), através da computação móvel, já é possível efetuar uma série de tarefas como enviar e receber faxes, emails, acessar bancos de dados, navegar na web, além do já disponível serviço de comunicação de voz nos conhecidos telefones celulares. Além disso, a computação móvel é vista como a combinação de três importantes e inter-relacionados elementos: Computação, Comunicação e Mobilidade, como pode ser visto na Figura 1. A comunicação é normalmente conduzida por redes sem fio, considerando possíveis desconexões, ruído, eco e baixa largura de banda. A mobilidade de alguns dispositivos podem transformar dados estáticos em redes estáticas a dados dinâmicos e voláteis em ambientes de redes sem fio. A estas características podemos ainda adicionar a baixa autonomia das baterias utilizadas nestes dispositivos, que também podem ocasionar desconexões ou um grande espaço de tempo permanecendo desligado.

Existem diferentes tipos de dispositivos móveis, dependendo de suas características e do montante que o usuário deseja investir na sua aquisição e manutenção. Alguns destes tipos são dispositivos muito simples com recursos bastante limitados. Neste caso, os dados são normalmente armazenados em computadores na rede cabeada e o dispositivo móvel faz o *download* dos dados quando necessário. Este cenário é bastante real para algumas aplicações e é atual-

mente o mais comum. Entretanto, o gerenciamento dos dados distribuídos não é severamente afetado pela mobilidade, pelo fato dos dados estarem armazenados normalmente em computadores na rede cabeada. Um ambiente interessante é caracterizado pela possibilidade de estações móveis armazenarem os dados de forma nativa e destes dados serem compartilhados por outras estações móveis. Alguns pesquisadores (IMIELINSKI; BADRINATH, 1993) chamam estas estações de *walkstations*.

A utilização de dados provenientes de servidores da rede estruturada em réplicas contidas nos dispositivos móveis, é uma forma interessante de se reduzir a quantidade de dados transferidos pela rede sem fio, diminuir a latência no acesso a estes dados, bem como a redução na utilização da largura de banda da rede disponível (XU; TANG; LEE, 2003; FIFE; GRUENWALD, 2003). Porém, isto só acontece se os dados que o usuário do dispositivo móvel necessita puderem ser encontrados no *cache* local. Por outro lado, é importante ressaltar que é freqüente a ocorrência de desconexões em ambientes de rede sem fio. Desta forma, torna-se difícil a verificação da validade dos dados armazenados localmente nos dispositivos móveis. Ainda assim, o armazenamento de dados no *cache* local promove um alto desempenho no processamento e uma alta mobilidade, permitindo que o uso da rede sem fio seja reduzido entre as unidades móveis e fixas, permitindo a utilização de dispositivos móveis de baixo custo e baixo investimento em infraestrutura de redes sem fio.

Segundo (CUNHA; DANTAS, 2004), a comunicação sem fio fornece suporte à computação móvel através de transmissões de dados via satélite, serviços de rádio, serviços móveis públicos[1], serviços para comunicação pessoal, entre outros. Essas tecnologias, associadas a equipamentos como PDA´s, permitem ao usuário um conjunto de serviços oferecidos em um sistema distribuído de computadores estacionários com mobilidade de acesso às informações.

Segundo (BARBARA, 1999), as unidades móveis podem ser agrupadas em sub-redes, onde cada uma destas sub-redes é gerenciada por uma estação de controle [2]. Cada ponto de acesso provê comunicação entre as unidades fixas ligadas a uma rede cabeada e as unidades móveis

[1] também conhecidos como *hot-spots*

[2] também chamada de estação de suporte móvel ou ponto de acesso (*access point*)

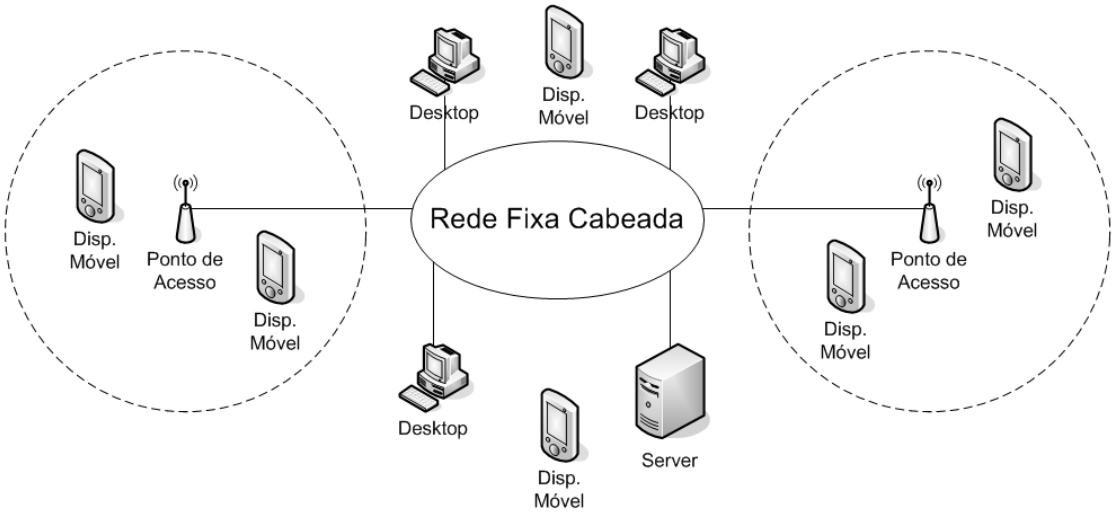

Figura 2: Um ambiente com suporte a computação móvel

que estão na área de cobertura dos pontos de acesso (Figura 2).

Uma rede sem fio com clientes móveis é essencialmente um sistema distribuído. No entanto, algumas características e limitações destes dispositivos nos levam a discussões interessantes para esta área de pesquisa. Nos ítens abaixo, serão apresentadas algumas destas características.

2.1.1 Portabilidade

De acordo com (CUNHA; DANTAS, 2004), um PDA é projetado para ser pequeno, leve, durável, operacional em uma variedade de condições e requerer o mínimo em termos de consumo de energia, para que o dispositivo possa permanecer o maior tempo possível fora de uma base fixa para carregamento desta bateria.

Segundo, (PITOURA; SAMARAS, 1998; BARBARA, 1999), seguem algumas dificuldades de projeto impostas pela portabilidade dos dispositivos móveis.

- Recursos limitados: Dispositivos móveis devem ser pequenos e leves, para serem facilmente levados de um local para outro, isso implica em possuir menos recursos em termos de processamento, memória e armazenamento persistente;
- Bateria limitada: Mesmo com avanços significativos em tecnologias para armazenamento

de energia, elas possuem capacidades relativamente limitadas e dependendo de sua localização, existe uma dificuldade para recarregar estas baterias. Além disso, a bateria é uma das maiores responsáveis pelo peso de um dispositivo móvel;

- Pouca robustez: Isto implica em uma facilidade maior em serem danificados e/ou roubados, além do acesso indevido aos dados nele armazenados. Por isto, o usuário deve ter cuidados maiores com a cópia de segurança dos dados armazenados e a proteção destes através de criptografia e senhas;

- Interface limitada: muitas vezes a interface com usuário está disponível através de pequenos teclados (em alguns casos canetas), e também pequenos visores, implicando muitas vezes, na redução das funcionalidades de aplicações desenvolvidas para estes dispositivos.

2.1.2 Mobilidade

A mobilidade de um dispositivo móvel se refere a sua localização em um ambiente. Muitas vezes, seus pontos de acesso à rede fixa são alterados conforme estes se movimentam, causando uma dinamicidade muito grande no ambiente. Esta mobilidade implica no desenvolvimento de novos métodos e técnicas, no que tange o gerenciamento distribuído dos dados nele armazenados. Assim, é difícil e complexa a tarefa de manter estas bases de dados atualizadas e consistentes. A mobilidade impõe restrições que eram inexistentes em ambientes de computação formados por computadores estacionários.

Tendo em vista estas mudanças de localização de elementos no ambiente, segundo (CUNHA; DANTAS, 2004), temos quatro subdivisões quanto a este critério:

- *nomadic computing*: o hardware pode se mover;

- *wireless computing*: o usuário pode se mover em um conjunto fixo de estações ligadas à rede;

- *mobile code/agent*: a aplicação pode se mover;

- *pervasive computing*: o usuário se move, executando aplicações móveis, sobre dados também móveis.

Conforme (PITOURA; SAMARAS, 1998), algumas conseqüências da mobilidade são:

- Configuração Dinâmica: os algoritmos tradicionais para processamento distribuído precisam ser reprojetados levando em conta a inexistência de uma topologia fixa da rede com os elementos móveis; a disseminação dos dados para os dispositivos móveis podem mudar dinamicamente;

- Localização: O custo para localizar os elementos móveis no ambiente está relacionado ao custo de comunicação que envolve cada um destes elementos. Partindo do princípio que a localização dos dispositivos pode mudar freqüentemente, é necessário que alguma estratégia seja adotada e implementada para consultar e gerenciar a localização dos mesmos;

- Heterogeneidade: A confiabilidade e o desempenho dos dispositivos móveis pode variar de acordo com sua localização e conectividade, e também de acordo com o montante que deseja-se investir em infraestrutura de comunicação e nos próprios dispositivos. Além disso, devido a variações de configurações e localização dos dispositivos, o número de serviços disponíveis pode variar.

2.1.3 Adaptabilidade

De acordo com (FISCHMEISTER; MENKHAUS; STUMPFL, 2003), para que possam continuar oferecendo serviços aceitáveis e reagir à mudanças conforme alterações no ambiente no qual se encontra um dispositivo móvel, é necessário que hajam mecanismos para coleta e gerenciamento das mudanças do ambiente. Nestes ambientes, pode haver uma mudança na taxa de transmissão e de erros de transmissão dos dados pela rede sem fio bem como alteração no tempo de latência no acesso à rede.

Segundo (NARAYANAN; SATYANARAYANAN, 2001), através de uma adaptação satisfatória, as aplicações que são executadas nos dispositivos móveis podem reagir de forma também satisfatória a estas alterações no ambiente. Em redes estacionárias, praticamente não há alteração nestes quesitos; isto já não ocorre em redes sem fio, que por sua natureza são altamente sujeitas a mudanças.

2.1.4 Gerência de Energia

Operações de leitura, escrita e comunicação podem ser interrompidas durante o período de desconexão do dispositivo móvel devido a restrições de energia proveniente de suas baterias. Estes dispositivos podem permanecer desconectados por longos períodos. Avanços nas tecnologias utilizadas para construção de baterias têm permitido que estes dispositivos permaneçam conectados por tempos maiores, mas inevitavelmente, sua energia é finita, sendo necessária uma recarga de tempos em tempos. Por estes motivos, segundo (FLINN; SATYANARAYANAN, 1999), é fundamental que a energia existente nas baterias seja aproveitada ao máximo, ou em outras palavras, desperdiçada ao mínimo.

Este assunto é tema de inúmeras pesquisas na área com o objetivo de minimizar o consumo de energia pelos dispositivos e maximizar seu tempo de autonomia. Como pode-se observar em (FLINN; SATYANARAYANAN, 1999), foi desenvolvida uma ferramenta denominada *PowerScope*, na qual é possível determinar o montante de energia consumido por cada aplicação. Em (OLSEN; NARAYANASWARNI, 2006), foi desenvolvido um esquema de gerenciamento de energia que trabalha em conjunto com o SO (Sistema Operacional) do dispositivo que pode estender o tempo de autonomia do mesmo em alguns casos em cerca de 42%. Já em (JIAO; HURSON, 2005), foram feitos alguns estudos para conservação de energia na utilização do dispositivo de comunicação 802.11b. Nestes estudos, verificou-se através do NS2 (*Network Simulator*), que ao atingir baixos valores de RTT (*Round Trip Time*), obteve-se uma economia de energia de até 65%.

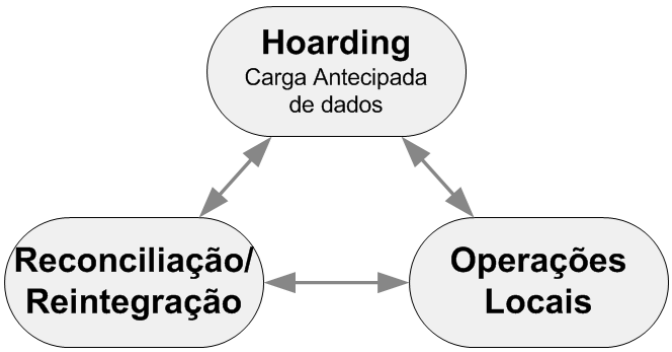

Figura 3: Estados das Operações de Desconexão

2.1.5 Fraca Conectividade

A fraca conectividade também é uma característica inerente à computação móvel. Podem existir momentos em que o dispositivo móvel permaneça com a conectividade limitada ou nula. Isto pode acontecer devido a variações na qualidade de sinal da rede sem fio ou mesmo na inexistência do mesmo pela ausência de estações de controle[3].

De acordo com (PITOURA; SAMARAS, 1998), operações de desconexão que ocorrem devido à fraca conectividade em ambientes de computação móvel envolvem três estados, como descrito abaixo e mostrado na Figura 3:

- **Hoarding**: pré-carregamento de ítens de dados no dispositivo móvel em forma de réplicas em armazenamento persistente local;
- **Operações Locais**: período em que o dispositivo móvel permanece desconectado da rede estruturada, podendo apenas realizar operações em seu cache local;
- **Reconciliação**: nesta operação, os dados existentes na unidade móvel são atualizados na(s) unidade(s) fixa(s). Isto deve ser tratado por uma semântica de serialização adequada, garantindo a não ocorrência de inconsistência nos dados atualizados.

Segundo (CUNHA; DANTAS, 2004), os problemas de perda total ou parcial de comunicação devem ser abrandados com a utilização de protocolos específicos. Podem também ser utilizados

[3]também conhecidas como pontos de acesso ou *access points*

mecanismos que visam a reintegração da conexão de forma tênue. Outra forma é a busca de informações em réplicas, identificando caso possível, qual das réplicas possui melhor disponibilidade para realizar a tarefa. Os principais protocolos que dão suporte a este tipo de problemas são:

- **Protocolo de Desconexão**: tem a finalidade de assegurar que o dispositivo móvel permaneça atuando sobre os dados durante um período de desconexão;
- **Protocolo de Desconexão Parcial**: possui a característica de dispor condições autônomas antecipadas para uma provável ocorrência de desconexão;
- **Protocolo de *Handoff***: tem a capacidade de retransmitir os dados pertencentes a um dispositivo móvel que está se deslocando de uma célula de cobertura para outra.

2.1.6 Interface Limitada

Conforme mencionado em (BARBARA, 1999), o tamanho da tela disponível nos dispositivos móveis fazem com que pesquisadores pensem em novas *interfaces* para recuperação de informações. A gerência de energia também está diretamente ligada às interfaces disponíveis.

Segundo (MUELLER; SCHAEFER; BLEUL, 2004), além do tamanho da tela, que normalmente é muito pequeno quando comparado a computadores fixos, alguns deles possuem teclados também bastante limitados. Alguns avanços foram feitos com relação às interfaces adicionando telas sensíveis ao toque (*touch-screen*) com a opção de realizar entradas através de canetas (*stylus*) e a dispositivos de entrada com reconhecimento de voz.

2.1.7 Replicação de Dados

A replicação de dados é comumente empregada para melhorar a disponibilidade de informações em sistemas distribuídos e é também uma importante técnica utilizada para permitir que um usuário possa utilizar uma aplicação em um dispositivo móvel, mesmo ele estando fora da área de cobertura de uma estação de controle. Assim, esta replicação pode melhorar o desempe-

nho e a disponibilidade das aplicações que fazem uso de dados provenientes da rede fixa (LOH et al., 2000).

De acordo com (WU; CHANG, 2006), um esquema de replicação determina o número e a localização das réplicas em tal ambiente. Esquemas de replicação tradicionais são estáticos no sentido de que o número e localização das réplicas é pré-determinado e fixo. Assim, esquemas de replicação tradicionais não são plenamente aplicáveis a ambiente de computação móvel.

O armazenamento persistente é um dos principais componentes de uma aplicação móvel. Esta tecnologia permite que o cliente móvel possa trabalhar com dados previamente armazenados no dispositivo que foram buscados através de uma conexão cabeada ou sem fio, sem a necessidade constante de estarem permanentemente conectados a rede estruturada (ITANI; DIAB; ARTAIL, 2005b).

As bases de dados que contém os ítens a serem replicados são chamadas de bases de dados consolidadas, que em sua grande maioria são provenientes da rede fixa. Já, as bases de dados que contém as réplicas dos dados são chamadas simplesmente de réplicas ou bases remotas.

A reconciliação é o processo inverso; após serem feitas atualizações na réplica, estas atualizações devem ser refletidas ou integradas na(s) base(s) consolidada(s), eliminando possíveis inconsistências (MADRIA; BHOWDRICK, 2001; LIM; HURSON, 2002). Além disso, é interessante que as mudanças efetuadas sejam propagadas para as outras réplicas existentes, como se pode observar na Figura 4. A propagação destas alterações podem ser realizadas através de diferentes canais ou protocolos de comunicação.

O procedimento de *hoard* é o acúmulo das mudanças realizadas no host móvel para serem propagadas às demais réplicas a fim de diminuir o volume de dados trafegados na rede, minimizando custos com comunicação de dados.

2.1.8 Capacidade de Armazenamento

A replicação de dados e o armazenamento persistente no dispositivo móvel estão diretamente relacionadas com a capacidade de armazenamento no dispositivo. Devido à capacidade

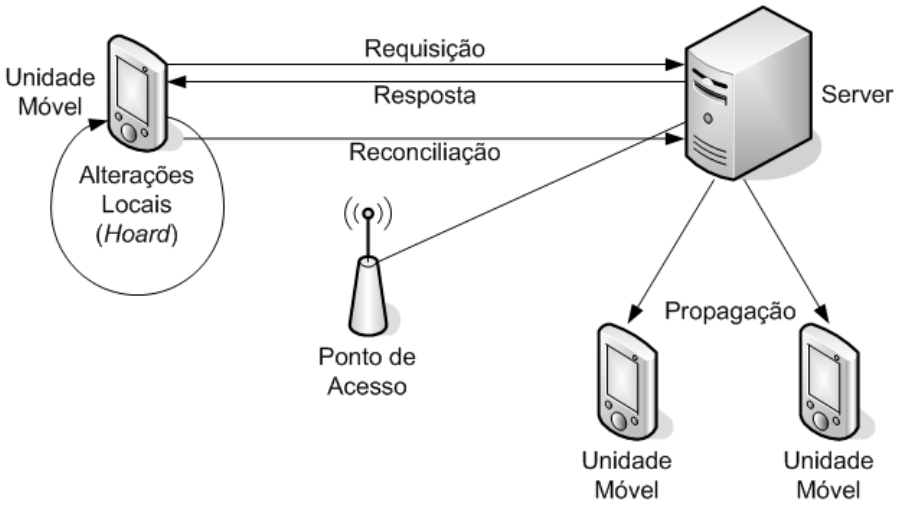

Figura 4: Modelo de Replicação, Reconciliação e Propagação de Dados

limitada de armazenamento existente em alguns modelos de dispositivos móveis, a criação e manutenção de réplicas de grandes bases de dados torna-se extremamente difícil (CHAN; RODDICK, 2003), particularmente em dispositivos como telefones celulares. Entretanto, a capacidade de armazenamento não está diretamente relacionada ao desempenho das aplicações.

2.2 Modelos de Comunicação em Computação Móvel

Em sistemas distribuídos tradicionais, existem diversos modelos de comunicação que podem ser utilizados. Contudo, em um ambiente móvel, devido às suas restrições, a utilização de alguns destes modelos fica prejudicada. Os modelos cliente-servidor, fim-a-fim e agentes móveis são apresentados a seguir como modelos que podem ser utilizados em ambientes de computação móvel.

2.2.1 Cliente-Servidor

De acordo com (RATNER; POPEK; REIHER, 1996), o modelo cliente-servidor, em sua forma mais simples, permite a desenvolvedores dividir a carga de processamento em dois processos lógicos: cliente e servidor. Nesta arquitetura, um cliente é geralmente uma aplicação que utiliza ou faz requisições de serviços ou dados provenientes de servidores, que por sua vez são apli-

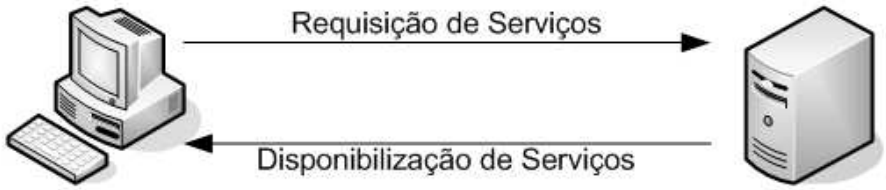

Figura 5: Modelo de Comunicação Cliente-Servidor

cações que fornecem estes serviços ou dados (Figura 5). A popularidade da Web fez com que esta arquitetura se tornasse uma das mais importantes aplicações deste modelo. Aqui utilizamos o termo aplicação, porque muitas vezes confunde-se cliente e servidor como sendo dispositivos específicos, quando na verdade é possível possuir o cliente e o servidor em um mesmo dispositivo, entretanto, na maioria dos casos estas aplicações estão em dispositivos diferentes.

Segundo (FEINSTEIN, 2000), o modelo cliente-servidor de três camadas é atualmente o método mais utilizado neste ambiente. As camadas são: *interface*, regras de negócio e banco de dados. A *interface* é responsável pela iteração do usuário com a aplicação, que pode fornecer informações em modos gráficos ou não, e gerencia a entrada de dados do usuário à aplicação. A camada de regras de negócio controla a execução da aplicação força o uso destas regras. A camada de banco de dados é responsável por gerenciar o acesso aos dados disponíveis nos meios permanentes de armazenamento através de SGBDs. Este modelo torna a aplicação menos frágil, isolando o cliente nas outras partes da aplicação, e também flexibiliza a mudança da aplicação, tanto do cliente, quando do próprio servidor.

Em um ambiente cliente-servidor tradicional, não é permitida a comunicação direta entre os clientes (RATNER; POPEK; REIHER, 1996). Assim, a comunicação entre eles, se necessário for, será sempre realizada através de um servidor. Além disso, dependendo do número de clientes, é importante ressaltar a possibilidade de replicação do serviço para atingir alta disponibilidade e performance na execução da aplicação.

2.2.2 Fim-a-Fim

Também conhecida como ponto-a-ponto ou P2P (*peer-to-peer*), ao contrário da arquitetura cliente-servidor, neste modelo existe a comunicação direta entre os clientes e ausência de servidores. Assim, cada dispositivo neste modelo de comunicação pode atuar tanto como cliente quanto servidor.

De acordo com (RATNER; POPEK; REIHER, 1996), soluções *peer-to-peer* permitem que os clientes comuniquem-se entre si, mas devido à mobilidade destes dispositivos o desempenho na comunicação está diretamente relacionada à qualidade do sinal disponível na rede *wireless*.

Segundo (FOX, 2001), a comunicação *peer-to-peer* tem conceitos vagos, entretanto, um item é comum a todas as definições: é orientada ao cliente. E uma de suas principais características é a colaboração.

Uma das mais conhecidas aplicações P2P existentes é o *Napster*, criado em 1999 por *Shawn Fanning*, da Universidade de *Northeastern*. Este sistema tinha por objetivo publicar os arquivos de áudio no formato MP3 armazenados do disco de cada cliente e também fazer o *download* de arquivos de outros clientes conectados à rede *Napster*. Além do *Napster*, existem dezenas de outros tipos de software para compartilhamento de arquivos pelos clientes, como *Imesh*, *eMule*, *MojoNation* que fazem uso de redes *Freenet* e *Gnutella*.

Além de softwares para compartilhamento de arquivos como os acima citados, existem outras categorias de software que fazem uso de comunicação P2P, tais como IM (*Instant Messenger*), jogos em rede, VoIP (Voz sobre IP (*Internet Protocol*)) e vídeo-conferência. Existem também várias tecnologias desenvolvidas que fazem uso de redes P2P. Entre elas podemos citar Jini e JXTA[4], que são plataformas para criação de redes colaborativas.

Outra característica de redes P2P é sua heterogeneidade. Praticamente qualquer dispositivo que possua uma *interface* de rede, rodando qualquer sistema operacional pode se conectar a uma rede deste tipo.

[4]Sun Microsystems

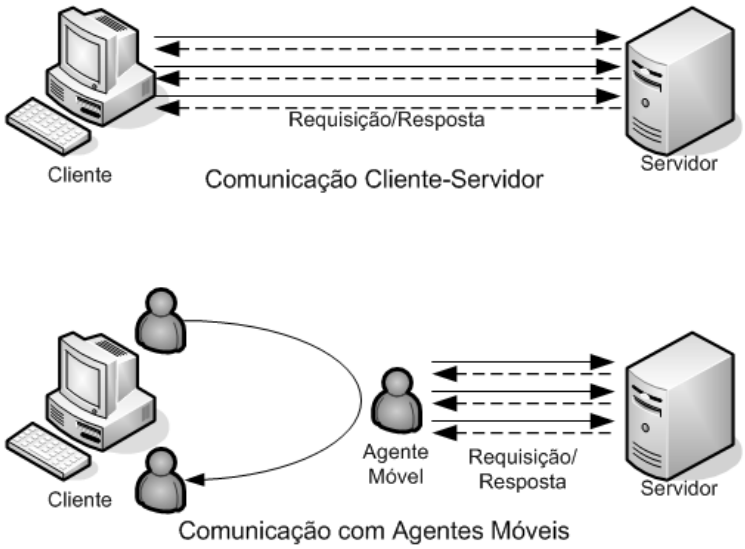

Figura 6: Agentes-Móveis podes otimizar o uso da largura de banda da rede.

2.2.3 Agentes Móveis

Segundo (PHAM; KARMOUCH, 1998), acredita-se que o paradigma de agentes móveis pode criar soluções atrativas para problemas de organização e recuperação eficiente de informações, necessidade de largura de banda e gerenciamento de redes. Ele afirma também, que o termo acima contém dois conceitos distintos: mobilidade a agentes. Pham e Karmouch também definem agentes móveis como sendo programas de computador auto-controlados que podem mover-se fisicamente pelos nós da rede, agir por iniciativa do usuário ou de outra entidade e reagir a eventos externos. A idéia de um programa auto-controlado sendo executado perto da fonte dos dados foi proposta para substituir o paradigma cliente-servidor como uma mais eficiente e flexível forma de comunicação, como mostrado na Figura 6.

Conforme já mencionado anteriormente, (GLITHO; OLOUGOUNA; PIERRE, 2002) ratifica que os agentes móveis são utilizados em aplicações que vão desde o gerenciamento de rede, distribuição automática de software, até recuperação de informações. Ele também menciona que o agendamento, escalonamento e sincronização dos agentes, pode se tornar um pesadelo, principalmente se um grande número de partes estiver envolvida.

De acordo com (KOTZ; GRAY, 1999), sistemas de agentes móveis diferem de sistemas de

migração de processo, porque no primeiro caso, o próprio agente determina para onde ele será transferido, ao contrário do conceito de migração de processo mais conhecida em sistemas distribuídos onde normalmente um escalonador efetua esta migração, normalmente para efetuar balanceamento de carga.

Agentes móveis também necessitam de um ambiente de execução especial, como (GLITHO; OLOUGOUNA; PIERRE, 2002) denomina de ambiente de execução de agentes. Estes ambientes fornecem algumas facilidades básicas tais como:

- Mobilidade: refere-se ao transporte ou migração do agente de um nodo para outro;
- Comunicação: diz respeito à comunicação do agente com o mundo externo;
- Identificação e Localização: os agentes móveis, assim como outras entidades, precisam ser nomeados. Além disso, é importante saber onde os agentes estão em um determinado momento;
- Segurança: os agentes móveis devem estar protegidos contra hosts e vice-versa. Atualmente, a maioria dos agentes móveis são implementados como aplicações Java que rodam na camada mais superior dos sistemas operacionais.

2.3 Redes Sem Fio

Segundo (DANTAS, 2002), os ambientes de redes sem fio (*wireless*) são configurações interessantes que podem agregar valor às redes locais de uma organização. O diferencial destes ambientes pode ser ilustrado pelo custo reduzido de sua infra-estrutura e o suporte de aplicações móveis. As WLANs (*Wireless Local Area Network*)[5] oferecem ganhos para os processos móveis envolvidos na utilização desta tecnologia, tais como a eficiência, precisão, e baixo custo da solução quando comparado com uma rede local que utiliza meios de transmissão guiados (cabos).

[5]Redes Locais Sem Fio

Diversos segmentos empresariais são naturalmente orientados a mobilidade, tais como: vendedores e inspetores de seguros, corretores imobiliários e de bolsa de valores, auxiliares de estoque e representantes comerciais. Desta forma, uma solução de rede móvel, com facilidade de tempo de resposta à solicitação de informações pode representar um diferencial de serviços para estes segmentos. Por outro lado, se algumas características já citadas anteriormente não forem observadas e consideradas no seu projeto, podem fazer com que a utilização das redes sem fio fiquem prejudicadas.

Uma rede sem fio estruturada é caracterizada por um *backbone* cabeado, ao qual estão conectadas estações de controle. Cada estação de controle coordena a comunicação de um ou mais dispositivos móveis na mesma célula, em outra célula, ou mesmo a um computador na rede cabeada. Em tal ambiente, os dados podem estar localizados na rede estruturada cabeada ou em dispositivos móveis.

Como se pode observar logo abaixo, existem diversas arquiteturas para suporte de utilização de dispositivos e aplicações móveis.

2.3.1 Breve Histórico

Segundo (MARIHART, 2001), as redes sem fio tiveram seu início por volta de 1980, quando a indústria desenvolveu sistemas de telefonia móvel. Neste período, os sistemas mais utilizados nesta primeira geração (1G) eram AMPS (*Advanced Mobile Phone Systems*), TACS (*Total Access Communication System*) e NMT (*Nordic Mobile Telephone*) e caracterizavam-se por serem redes analógicas e voltadas para comunicação de voz.

A segunda geração (2G) caracteriza-se por dispositivos inteiramente digitais e por melhorarem substancialmente a qualidade de sinal disponibilizada pelas redes 1G. A 2G utiliza sistemas tais como: GSM (*Global System for Mobile Communication*), TDMA (*Time Division Multiple Access*) e CDMA (*Code Division Multiple Access*). Contudo, estas redes não satisfazem a necessidade de alta velocidade para transmissão de dados.

A terceira geração (3G) é uma evolução natural da 2G em termos de velocidade e servi-

ços, padronizada pelo ETSI (*European Telecommunications Standard Institute*) e representa o UMTS (*Universal Mobile Telecommunication System*). Possui recursos ou serviços tais como banda-larga, multimídia e transmissão de vídeo em tempo-real.

2.3.2 WAP (*Wireless Application Protocol*)

De acordo com (DANTAS, 2002), WAP é uma especificação de arquitetura de protocolo voltada para atender usuários de dispositivos móveis que necessitam de acesso à Internet. Em outras palavras, pode-se dizer que WAP é uma arquitetura de protocolo que permite que dispositivos móveis executem aplicações através de redes sem fio, fazendo acesso aos serviços disponíveis no ambiente da Web.

Dentre um vasto número de serviços disponíveis para usuários WAP, podemos citar acesso a notícias, previsão do tempo, cotação de ações e *internet banking*. Contudo, para prover estes serviços de maneira satisfatória, alguns desafios como segurança, desempenho, retardo de acesso à rede e largura de banda pequena, devem ser transpostos.

Como características principais do WAP, temos:

- utilização de um modelo de programação semelhante ao da Web;
- utilização da WML (*Wireless Markup Language*), desenvolvida para ser utilizada em dispositivos com pouca memória, em substituição ao HTML (*Hyper Text Markup Language*), utilizado em grande parte dos *web sites*;
- utilização da linguagem WMLScript (*Wireless Markup Language Script*), em substituição ao JavaScript;
- utilização do WTA (*Wireless Telephony Application*) e WTAI (*Wireless Telephony Application Interface*), que caracterizam um conjunto de extensões específicas para serviços de voz.

As principais diferenças entre um ambiente de Internet convencional e o ambiente WAP,

são que o segundo deve garantir uma alta taxa de compressão na transmissão dos dados e uma tentativa de aproveitar ao máximo a pequena largura de banda disponível. Razões para este fato já foram apresentadas anteriormente.

Com relação ao modelo e arquitetura de protocolos que fazem parte do WAP temos:

- WAE (*Wireless Application Environment*): ambiente para aplicações de modo geral baseado em uma combinação da WWW (*World Wide Web*) e tecnologias de telefonia móvel;
- WSP (*Wireless Session Protocol*): responsável pelas operações remotas entre um cliente e um *proxy* ou servidor. Foi projetado para dar suporte a serviços orientados e não-orientados a conexão;
- WTP (*Wireless Transaction Protocol*): definido para prover os serviços necessários para aplicações interativas de *browsing* (requisição-resposta) de forma confiável;
- WTLS (*Wireless Transport Layer Security*): desenvolvido para prover segurança no transporte baseado em outro padrão conhecido como TLS (*Transport Layer Security*), anteriormente conhecido como SSL (*Secure Socket Layer*);
- WDP (*Wireless Datagram Protocol*): comunicação não-orientada a conexão e fornecem suporte para camadas superiores como WTLS, WTP e WSP.

2.3.3 *Bluetooth*

Segundo (DANTAS, 2002), esta tecnologia é um padrão de fato com uma especificação aberta para enlaces entre dispositivos móveis tais como computadores pessoais, PDAs, telefones celulares e outros dispositivos portáteis de baixo custo usando ondas de rádio de curto alcance.

Sua proposta é habilitar o usuário para a conexão de uma grande variedade de equipamentos de maneira simples e fácil. A operação do *Bluetooth* ocorre em uma banda de freqüência entre 2.402 e 2480 GHz e possui compatibilidade mundial. Outro item importante a acrescentar, é que a especificação da tecnologia *Bluetooth* é aberta, significando que qualquer empresa pode obtê-la para desenvolvimento de seus produtos.

Equipamentos dotados desta tecnologia carregam um *chip* capaz de conectar-se automaticamente a outros dispositivos que também possuem esta característica, por meio de ondas de rádio. Esta tecnologia permite por exemplo, transmissão de voz em tempo real e podem ter o alcance de 10 até 100 metros (dependendo da configuração de potência utilizada).

Segundo (MCDERMOTT-WELLS, 2004), esta tecnologia, embora originalmente desenvolvida como uma substituição a sistemas cabeados desenvolvida pela Ericsson (uma das maiores fabricantes de telefones celulares) em 1994, esta capacidade desta tecnologia está se difundindo em inúmeros tipos de dispositivos, tais como PDAs, telefones celulares, computadores pessoais e portáteis, periféricos diversos (mouses, teclados, joysticks, câmeras, impressoras, pontos de acesso a redes sem fio, fones de ouvido, alto-falantes, *stereo-receivers*, dispositivos automotivos, sistemas industriais, instrumentos musicais entre outros.

2.3.4 WLANs e o Padrão IEEE 802.11

De acordo com (PROMMAK et al., 2002), as WLANs tais como aquelas confeccionadas de acordo com o padrão 802.11, estão sofrendo um grande crescimento, provendo funcionalidades de rede que complementam dispositivos tais como telefones celulares. Desse modo, *notebooks* e PDAs estão ficando cada vez mais acessíveis, menores e poderosos, também demandando a utilização de redes sem fio.

Como descrito por (FERRO; POTORTI, 2005), também conhecido como Wi-Fi (*Wireless Fidelity*), o IEEE 802.11 compreende um conjunto de padrões utilizados para a implementação de WLANs. A família 802.11 inclui seis técnicas de modulação que usam o mesmo protocolo, sendo que os mais conhecidos são 802.11a, 802.11b e 802.11g. Os dois últimos usam a banda de 2,4GHz, enquanto o primeiro usa a banda de 5GHz. Com relação ao *throughput*, o primeiro e o último atingem a taxa de 56Mbps, enquanto o segundo atinge a taxa de 11MBps.

Este padrão define procedimentos MAC (*Media Access Control*) para acesso à camada física, que pode ser através de infra-vermelho ou freqüências de rádio. A Figura 7 mostra a pilha de protocolos do padrão 802.11. A mobilidade é tratada na camada MAC, desta forma, a

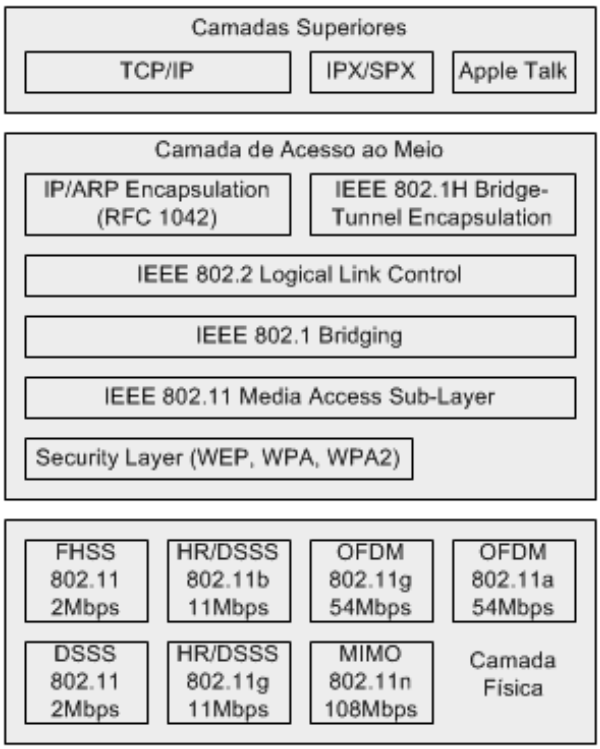

Figura 7: Pilha de Protocolos do Padrão 802.11

mudança de uma célula para outra é transparente para as camadas superiores de um dispositivo com esta funcionalidade.

O estabelecimento do padrão 802.11, pelo IEEE, foi um dos fatores que ajudaram a reverter o quadro de falta de padronização no uso de tecnologias *wireless*, proporcionando convergência e interoperabilidade entre implementações proprietárias. A topologia 802.11 tem por meta a integração transparente dos elementos móveis de uma LAN (*Local Area Network*) com relação aos níveis superiores da pilha de protocolos.

Além disso, o IEEE 802.11 suporta duas topologias, como seguem:

- IBSS Networks (*Independent Basic Service Set*)[6]: também chamada apenas de BSS, esta topologia é caracterizada pela inexistência de uma estrutura de backbone e ser constituída de pelos menos dois dispositivos wireless. Uma rede deste tipo é comumente chamada de rede *ad-hoc* (rede temporária), pois pode ser montada rapidamente sem um planejamento e sem pontos de acesso, como se pode observar na Figura 8.

[6]Rede de Conjuntos de Serviços Básicos Independentes

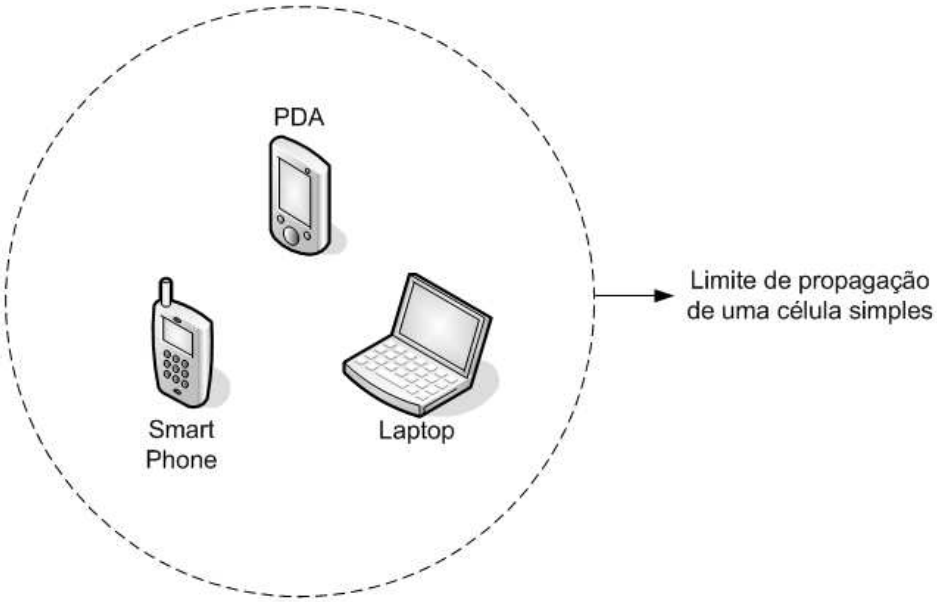

Figura 8: Exemplo de Rede BSS/*Ad-Hoc*

- ESS Networks (*Extended Service Set*)[7]: estas redes compreendem uma maior área de abrangência e maior complexidade, motivada pela extensão das redes BSS. Um exemplo destas redes já foi mostrado na Figura 2.

Atualmente, segundo (LORINCZ; BEGUSIC, 2006), através da especificação 802.11g, já é possível a utilização de taxas de transferência da ordem de 54 Mbps (*Megabits per second*). Entretanto, cerca de 50% a 60% é disponível para o usuário devido ao *overhead* imposto pelo cabeçalho do quadro da camada física, por preâmbulos e pela necessidade de se obter o ACK (Acknowledge), de cada quadro. Mesmo assim, é possível executar aplicações que exigem banda larga, incluindo aplicações multimídia.

Recentemente, o IEEE anunciou que formou um novo "802.11 Task Group N"para desenvolver novos adendos ao padrão para WLANs. O objetivo do grupo é atingir a taxa de transferência de 100 Mbps, através da subtração do *overhead* para gerenciamento de funcionalidades do protocolo como as já citadas no parágrafo anterior.

[7]Rede de Conjuntos de Serviços Estendidos

2.3.5 Redes Ad-Hoc e MANET

As MANETs (*Mobile Ad-Hoc Networks*), são uma emergente área de pesquisa. Segundo (FIFE; GRUENWALD, 2003), a maioria dos trabalhos relacionados às MANETs discutem sobre protocolos de roteamento. No caso desta dissertação, o objetivo de inserir as MANETs neste capítulo é com relação à comunicação de dados nestes ambientes.

Como já mencionado anteriormente, uma rede sem fio tradicional, é constituída por uma rede fixa cabeada, na qual estão conectados os pontos de acesso, e os dispositivos móveis fazendo acesso aos recursos disponíveis através destes pontos de acesso.

Dentro das redes sem fio tradicionais, os servidores têm fornecimento permanente de energia. Os dispositivos móveis podem comunicar entre si através de pontos de acesso. Entre estes fatos neste tipo de rede, podemos incluir o consumo de energia pelos dispositivos, conectividade da rede, e conseqüente comunicação com o servidor.

Porém, uma MANET é um conjunto de clientes e servidores móveis. Neste caso, todos os nós se comunicam através de uma interface *wireless*, são móveis e energizados por baterias. A topologia também pode mudar freqüentemente, onde os nós se organizam automaticamente e podem formar uma rede isolada ou anexada a uma rede maior, como a Internet. Complementando, todos os nós podem se comunicar livremente uns com os outros e esta tecnologia tem uso prático sempre que uma rede sem fio tradicional não estiver presente.

Alguns autores tais como (CORSON; FREEBERSYSER; SASTRY, 1999; KAHN; KATZ; PISTER, 1999; LIU et al., 2002), afirmam que esta tecnologia é uma área de pesquisa interessante para uso militar, operações de resgate e sensores. O suporte a estas aplicações necessita normalmente da presença de um banco de dados para armazenar e transmitir informações críticas, tais como inventários e informações táticas.

Outro item considerado crucial por (COATTA et al., 2004) é que redes sem fio tradicionais envolvem servidores em praticamente todas as comunicações, enquanto as MANET incluem a possibilidade de receber e enviar dados aos servidores, mas também permitem que os clientes

comuniquem entre si diretamente sem envolver necessariamente um servidor.

Segundo (XU; HISCHKE; WALKE, 2003), os nós em uma MANET podem ser classificados quanto às suas capacidades. Um cliente ou SMH (*Small Mobile Host*) é um nó com capacidade reduzida de processamento, armazenamento, memória, comunicação e energia. Exemplos de SMH são telefones celulares e alguns modelos de PDA e *smart phones*. Um servidor ou LMH (*Large Mobile Host*), possui capacidade maior e exemplos desta classificação de dispositivo móvel são *notebooks* e *laptops*.

Os servidores, devido a sua capacidade maior, podem possuir SGBD (Sistemas Gerenciadores de Banco de Dados) completos, e podem ser responsáveis por operações de *broadcast* e pelo envio de dados para os clientes. Já os clientes podem armazenar partições dos bancos de dados presentes nos servidores em seu armazenamento persistente.

Os nós de uma MANET podem não permanecer conectados durante todo o tempo. Para estar conectado à rede, é necessário que ele esteja ao alcance de no mínimo, outro nó e que possua energia suficiente para funcionar. Mesmo assim, um nó de uma MANET pode se comunicar com um nó de outra MANET, mesmo não estando na área de alcance do sinal da rede sem fio, através de protocolos de roteamento específicos (EHSAN; UZMI, 2004)[8], conforme podemos ver na Figura 9.

2.3.6 Modos de Operação

Enquanto que em um sistema distribuído tradicional um *host* opera em apenas dois modos: totalmente conectado e totalmente desconectado, os dispositivos móveis, conforme a condição em que se encontram, podem assumir alguns modos de operação, que segundo (PITOURA; BHARGAVA, 1994), são:

- *Fully Connected*: ou totalmente conectado, é o modo de operação no qual o dispositivo móvel encontra-se em pleno funcionamento;

[8]normalmente os LMH possuem uma área de alcance do sinal da rede sem fio maior que os SMH e também podem permanecer ligados por mais tempo

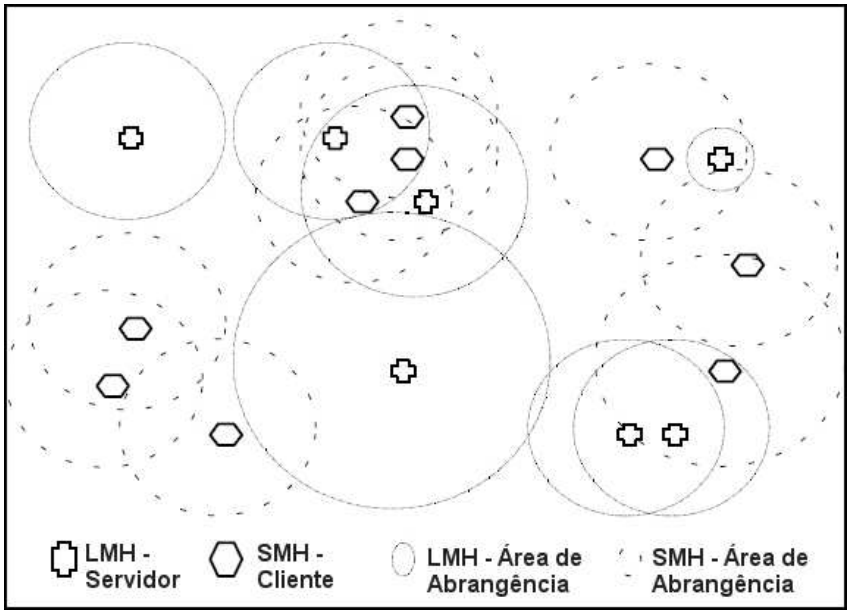

Figura 9: Nós de uma MANET geograficamente distribuídos
Fonte: (FIFE; GRUENWALD, 2003)

- *Partially Connected*: ou parcialmente conectado, refere-se ao modo de operação no qual a conexão à rede sem fio é presente, porém limitada;
- *Disconnected*: ou desconectado, significa que a disponibilidade da rede sem fio é nula;
- *Doze Mode*: ou modo de cochilo, representa o modo de conservação de energia, no qual sua velocidade é reduzida ou o *display* do dispositivo encontra-se desligado. O dispositivo pode ser "acordado" pela intervenção do próprio usuário ou pelo recebimento de uma mensagem externa;

A Figura 10 representa a relação entre estes modos de operação. Segundo (GUPTA; SRIMANI, 2001), de forma que a maioria das transições entre os modos ocorra de modo previsível, alguns protocolos podem ser projetados para preparar o sistema para a transição entre eles.

- Protocolo de Desconexão: é executado quando o dispositivo móvel é fisicamente desconectado da rede fixa. Este protocolo deve assegurar que a informação necessária deve estar disponível localmente para operar durante a desconexão, bem como notificar os demais dispositivos que a desconexão irá ocorrer;

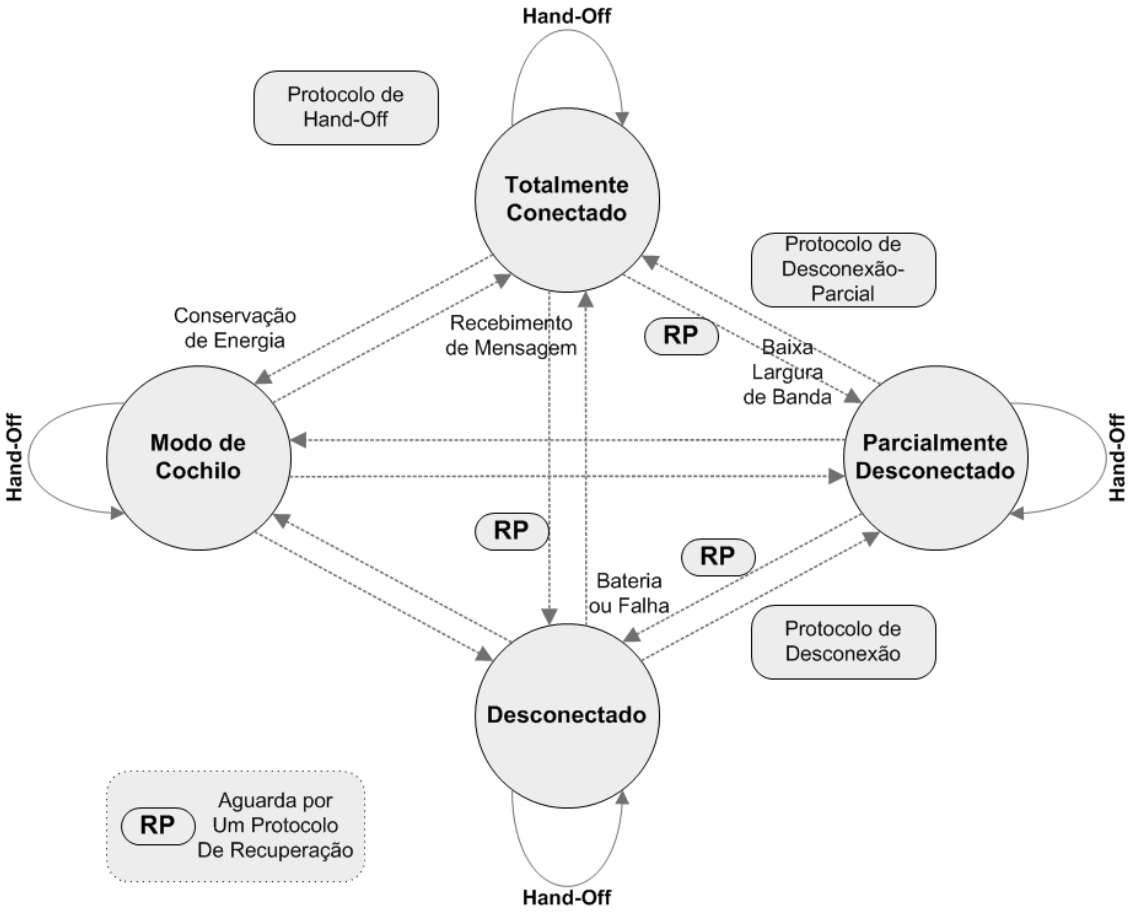

Figura 10: Estados de Operação de um Dispositivo Móvel
Fonte: (PITOURA; BHARGAVA, 1994)

- Protocolo de Desconexão Parcial: prepara o dispositivo para operação em um modo no qual todas as conexões à rede fixa devam ser restritas. A criação de um cache seletivo deve ser feita para minimizar usos futuros da rede;

- Protocolos de Recuperação: restabelecem a conexão com a rede fixa e continua normalmente as operações;

- Protocolos de *Hand-Off*: referem-se à mudança do dispositivo de uma célula para outra. O modo de operação do dispositivo deve ser transferido para a estação de controle da nova célula.

2.3.7 Sincronização e Atualização de Relógios

O padrão IEEE 802.11 suporta o modelo de comunicação ponto a ponto através do IBSS, que é uma rede ad-hoc composta por nodos que podem se comunicar livremente. Na maioria dos sistemas distribuídos e também em redes sem fio, a sincronização dos relógios possui fundamental importância. Este é um fator chave para prover um efetivo gerenciamento de energia e suportar o controle do acesso ao meio de transmissão. Além disso, é empregado no tratamento de QoS (*Quality of Service*) em redes *ad-hoc*, particularmente em aplicações de tempo-real.

A natureza dinâmica das redes *wireless ad-hoc*, o não determinismo do canal de comunicação sem fio e a falta de nodos de referência na rede fazem com que a sincronização de relógios nestes ambientes seja considerada desafiadora (CHEN; LENEUTRE, 2006). Um mecanismo ideal de sincronização de relógios para redes *ad-hoc*, devem ser robustos à variação de topologia e mobilidade, eficientes em termos de custo de transmissão e tráfego, escalável e seguro.

A sincronização de relógios em uma rede de computadores tem por objetivo fornecer uma escala de tempo comum a todos os nós que fazem parte desta rede. De modo que todos os relógios de hardware são imperfeitos, os relógios locais podem estar diferentes dos demais em um determinado tempo (BEEK et al., 1999). Esta diferenciação pode ser notada, observando-se a duração ou o momento em que determinada operação é executada.

Entretanto, para diversas aplicações ou protocolos de rede, é necessário que uma visão única do tempo exista e esteja disponível para todos ou parte dos nós em um instante particular (MOSTOFI; COX, 2004). O protocolo NTP (*Network Time Protocol*), foi projetado para distribuição de informações sobre data/hora em um grande e heterogêneo ambiente, tal como o da internet, onde as taxas de transmissão podem variar bastante.

De acordo com (MILLS, 1994), este protocolo utiliza uma arquitetura simétrica na qual uma sub-rede de servidores de hora operam em conjunto de forma hierárquica e auto-organizada, que por sua vez efetuam sua atualização em relógios atômicos. Estes servidores também podem redistribuir esta informação de hora em uma rede local através de algoritmos de roteamento e serviços de tempo/hora.

3 *Coerência, Consistência, Replicação, Particionamento, Disseminação e Reconciliação de Dados*

Em ambientes de computação móvel, múltiplos dispositivos requisitando os mesmos ítens de dados podem causar conflitos. Além disso, os dispositivos móveis podem permanecer desconectados da rede sem fio-estruturada por longos períodos de tempo. Estas desconexões podem ocorrer devido à falta de energia nas baterias, ou pela saída do dispositivo da área de cobertura das estações de controle. Os dispositivos móveis podem também ser realocados para outras células para se comunicarem com diferentes servidores de dados ou para executar aplicações diferentes. Neste sentido, um método interessante para gerenciamento do *cache* em ambientes de rede sem fio deve ser capaz de lidar com problemas relacionados à limitação de recursos e suas freqüentes desconexões (XU et al., 2004).

O armazenamento local das informações freqüentemente acessadas é uma importante técnica para reduzir a disputa no canal de comunicação entre cliente e servidor (BARBARA; IMIELINSKI, 1995; CAI; TAN, 1999). Em adição, este armazenamento pode ser uma forma de economizar energia das baterias, de modo que não é necessário gastá-la com o envio e recepção de dados na rede sem fio. O armazenamento dos dados nos dispositivos móveis podem ainda representar economia de fundos, dependendo do custo de criação e de manutenção de redes sem fio estruturadas.

Entretanto, considerando que os dados são armazenados localmente nos dispositivos móveis, é necessário o uso de mecanismos para prover a consistência destes dados. Estes meca-

nismos são aplicados de forma que seja possível se certificar que os dados enviados para os clientes são consistentes com os dados armazenados no servidor. Levando em conta que os dispositivos móveis podem permanecer desconectados ou ficar sem energia em suas baterias por longos períodos de tempo, esta verificação de consistência se torna mais complexa.

3.1 Coerência de *Cache*

No contexto da computação móvel, pode-se definir *cache* como uma forma de armazenar os dados mais relevantes no dispositivo local, eliminando ou reduzindo a necessidade de acessar remotamente dispositivos de armazenamento e desta forma, aumentando o desempenho das aplicações que fazem uso de dados provenientes de dispositivos remotos através da rede. Assim, pode-se efetuar operações locais no dispositivo móvel de forma mais rápida e consumindo uma menor quantidade de energia. Segundo (XU et al., 2004), esta solução é considerada eficiente para lidar com a ineficiência da rede sem fio porque reduz o tráfego de dados pela rede.

Em adição, (XU et al., 2004) informam que existem três importantes fatores que devem ser considerados no gerenciamento de *cache* no cliente:

- Uma política de substituição do *cache* determina quais ítens de dados devem ser eliminados do *cache*, quando há espaço insuficiente para acomodar novos ítens de dados;
- Uma política de busca antecipada pré-carrega ítens de dados no *cache* para possíveis futuras requisições de dados;
- Um esquema de invalidação de *cache*, mantém a consistência dos dados entre o cliente e o servidor.

Segundo (BERKENBROCK; DANTAS, 2005), dependendo da aplicação, para manter a coerência de *cache* no dispositivo móvel de forma efetiva e confiável, existem duas técnicas comumente utilizadas em aplicações móveis que são usualmente somente para leitura e nas quais os dados sempre são fornecidos a partir de servidores estacionários. São elas:

- a notificação de invalidação de registros, técnica utilizada quando o volume de dados alterados no servidor corresponde a menos de 50% da massa de dados compartilhada, para reduzir o volume de dados transferidos pela rede sem fio;
- a notificação de validação de registros, técnica utilizando quando o volume de dados alterados no servidor corresponde a mais de 50% da massa de dados compartilhada.

De forma que normalmente menos de 50% da massa de dados é alterada, as notificações de invalidação são bem mais comuns do que as notificações de validação. Desta forma, no trabalho de (BERKENBROCK; DANTAS, 2005), são citadas algumas abordagens para mecanismos de notificações de invalidação, como seguem:

- ***Stateful***: Nesta abordagem o servidor tem conhecimento de quais unidade móveis encontram-se em sua célula, assim como possui conhecimento do estado do *cache* das suas unidades. Devido a estes fatores, grande parte dos trabalhos realizados em coerência de *cache* são baseadas na abordagem seguinte (*stateless*);
- ***Stateless***: Ao contrário da anterior, nesta abordagem o servidor não possui conhecimento de quais elementos estão em sua área de cobertura, nem do estado do *cache* armazenado nestes elementos. Nesta abordagem o servidor mantém um histórico das atualizações realizadas e fornece estas informações para os dispositivos móveis através de *broadcasts*, ou quando o dispositivo efetuar uma solicitação explícita.

Ainda segundo (BERKENBROCK; DANTAS, 2005), ao longo dos anos foram concebidas algumas estratégias para invalidação de *cache* como as citadas abaixo:

- ***Broadcast Timestamp*(TS)**: nesta estratégia o servidor envia um broadcast contendo uma notificação de invalidação para os clientes, informando os ítens de dados que foram alterados nos últimos ω segundos. Esta notificação contém o *timestamp* da notificação atual (T_i) e uma lista de tuplas (j, tj), onde j especifica o ítem de dado que foi alterado e tj indica quando a alteração foi realizada (BARBARA; IMIELINSKI, 1995).

A unidade móvel mantém uma variável T_l indicando quando a última notificação foi recebida. Se a diferença entre T_l e T_i for maior do que ω, então todo o *cache* da unidade móvel é apagado. Caso o ítem contido na notificação de invalidação tenha sido alterado em um tempo maior que o *timestamp* armazenado no *cache*, a unidade apaga este ítem do seu *cache*. Assim, se o cliente precisar daquele dado que foi excluído, este deve solicitá-lo novamente ao servidor. É importante mencionar aqui a importância da sincronização dos relógios dos dispositivos móveis, conforme apontado por (CHEN; LENEUTRE, 2006; MOSTOFI; COX, 2004).

A unidade móvel também mantém uma lista Q_i, com informações consultadas no intervalo [T_{i-l},T_i]. Por fim, o *timestamp* do *cache* do cliente móvel (T_l) será alterado de acordo com o *timestamp* atual (T_i);

- ***Cache Coherence Schema with Incremental Update Propagation*(CCS-IUP)**: esta estratégia visa evitar a invalidação dos dados do *cache*, reduzindo o número de perdas de informações. Além disso a técnica propaga apenas as partes relevantes das modificações que afetam o dado armazenado de forma persistente no dispositivo móvel. Isto reduz significativamente a quantidade de dados transmitidos (CHUNG; CHO, 1998).

  Na estratégia em questão, o banco de dados é representado por um conjunto de relações (R_i). Cada relação possui um número n de registros. Para cada registro na relação R_i, existe um identificador único RID (*Record IDentificator*). O servidor deve enviar um *broadcast* periódico no tempo T_k com uma notificação de invalidação contendo informações sobre os registros alterados nos últimos Δ segundos.

  A notificação de invalidação é definida formalmente como uma lista ϕ cujo elemento é uma 4-tupla (R_i,RID,VAL,TS), sendo que R_i representa o identificador da relação; RID representa o identificador do registro que foi alterado no intervalo [$T_k - \Delta$,T_k]; VAL representa o novo valor do registro e TS representa o *timestamp* indicando quando o registro foi alterado pela última vez.

- ***Amnestic Terminals*(AT)**: nesta estratégia o servidor fica responsável por informar os

identificadores dos ítens de dados que foram alterados desde a última notificação de invalidação (BARBARA; IMIELINSKI, 1995).

A notificação de invalidação contém uma lista com os identificadores dos dados alterados(j). Após receber a notificação, o dispositivo móvel compara o ítem recebido com o ítem correspondente no seu *cache*. Se um ítem de dado armazenado na unidade móvel for informado na notificação, o dispositivo apaga este ítem do seu *cache*. Caso contrário a unidade móvel considera o item do seu *cache* válido. Nesta estratégia a unidade móvel também mantém uma lista Q_i com informações consultadas no intervalo [T_{i-l},Ti];

- ***Grouping with COld update-set REtention*(GCORE)**: De acordo com (WU; YU; CHEN, 1996), esta estratégia visa reduzir o número de invalidações desnecessárias. Nela, o servidor particiona o banco de dados em um número de grupos. Desta forma, o servidor consegue dinamicamente identificar os ítens de dados *hot* e *cold* em cada grupo. Os ítens de dados *hot* são aqueles que foram incluídos na mais recente notificação de invalidação enviada para as unidades móveis. Ao invés de examinar todo o grupo, o GCORE exclui os ítens de dados *hot* do grupo quando verifica a sua validade. Com as alterações *hot* excluídas do grupo, o servidor pode concluir que os objetos que não foram alterados no grupo podem ser retidos no *cache*.

No presente trabalho de pesquisa, os dados podem ser fornecidos pelo servidor estacionário, mas também podem ser recebidos de outros clientes, que também podem atuar como servidores móveis. De modo que uma abordagem pessimista está sendo utilizada e os servidores não sabem quais os dados que estão armazenados nos dispositivo móvel, todos os dados requisitados são transferidos através da rede. Assim, será utilizado o mecanismo de timestamp para manter a consistência dos dados armazenados localmente nos dispositivos, de forma que é uma técnica amplamente utilizada em sistemas de gerenciamento de banco de dados distribuídos (OZSU; VALDURIEZ, 1999).

3.2 Consistência

A replicação dos dados nos dispositivos móveis gera múltiplas cópias de uma mesma informação. Este fato introduz o problema de consistência entre as cópias. Para prover um mecanismo de confiabilidade nas réplicas, utiliza-se das propriedades conhecidas como ACID (Atomicidade, Consistência, Isolamento e Durabilidade).

- **Atomicidade**: esta propriedade determina que as transações executadas em bancos de dados sejam atômicas, em outras palavras, indivisíveis. Ela garante que todas as operações sejam refletidas corretamente ou nenhuma delas o será. Não existe divisão de transações em partes menores, desta forma, uma transação é vista como uma ação única. Se uma transação é executada até o final sem nenhum problema, seus resultados serão efetivados, caso contrário, todas as operações incluídas na transação serão abortadas;

- **Consistência**: esta propriedade determina que após a execução de uma transação, a consistência do banco de dados deve ser preservada; em outras palavras, significa que mesmo no caso do processamento concorrente de várias transações, o resultado obtido será o mesmo que seria obtido através da execução seqüencial das transações;

- **Isolamento**: esta propriedade refere-se à serialização das transações concorrentemente executadas. O sistema deve garantir que a execução de uma transação não faça sua atualização visível para outras transações até o final de sua execução, de forma que o usuário tenha a percepção de que uma transação seja executada após a outra. Desta forma, a execução concorrente deve ser equivalente à execução serial das mesmas;

- **Durabilidade**: esta propriedade determina que depois de executada uma transação com sucesso, as mudanças que esta acarreta ao Banco de Dados persistem, mesmo havendo falhas no sistema.

3.3 Replicação

Segundo (OZSU; VALDURIEZ, 1999), a replicação de banco de dados é o processo no qual é possível efetuar o compartilhamento de dados entre bancos de dados distribuídos. Através de mecanismos específicos, cópias do banco de dados chamadas de réplicas podem ser distribuídas em outros nós da rede com o propósito de aumentar a velocidade no acesso aos dados, bem como de aumentar a disponibilidade da informação. Estes dados, depois de serem submetidos a mudanças, são reintegrados ao servidor por meio de primitivas de sincronização.

Uma característica indispensável em bancos de dados, quando utilizados em ambientes de mobilidade, é a habilidade de lidar com conexões sujeitas constantemente à interferência do ambiente e a ausência ou variação de qualidade de sinal. Conforme escrito por (CHEN, 2003), apesar de aspectos como grau de divisão, localização e fatores de integridade, a utilização de réplicas tem sido comum.

Conforme (CUNHA; DANTAS, 2004), a criação e manutenção das réplicas em bancos de dados móveis seguem fases distintas, conforme os modos de operação de um dispositivo móvel anteriormente apresentados (preparação para desconexão, operações em modo de desconexão e re-conexão), conforme mostrado na Tabela 1. Durante o período em que uma unidade móvel esteja conectada a uma unidade fixa ou que possua boa conexão à rede sem fio, ela deve ser preparada para um possível período de desconexão. Nesta etapa são determinados quais dados serão armazenados na memória do dispositivo portátil. De acordo com o plano, podem ser adotadas as seguintes medidas:

- A definição dos dados que serão copiados para o dispositivo fica a critério da necessidade do usuário;
- A medida que um usuário executa operações de consulta e atualização nos dados locais, o sistema monitora estas ações e armazena dados antecipadamente;
- Os dados são encaminhados para a unidade móvel de acordo com o nível de permissões do usuário.

	Desafio	Plano
Preparação para desconexão	O que armazenar na memória	- Especificado pelo usuário - Baseado no histórico das operações - Definido pela aplicação
	Quando armazenar na memória	- Antes da desconexão - Em um período definido
Operações em modo de desconexão	Requisição de dados não armazenados	- Criação de Exceções - Fila para futuros serviços
	O que guardar no log	- Dados - Operações - *Timestamps*
	Como usar o log	- De forma incremental - Associado ao banco de dados
Reconexão	Como integrar registros	- Submeter todos os dados - Re-executar o log
	Como resolver conflitos	- Uso de semânticas de aplicações - Soluções automáticas - Uso de especificações

Tabela 1: Planos para Geração e Tratamento de Réplicas. Fonte: (CUNHA; DANTAS, 2004)

Além de saber quais ítens de dados deverão ser submetidos ao dispositivo móvel (ou requisitados por ele), também é necessário que seja definido qual o instante mais indicado para o recebimento destes. Neste ponto, a disponibilidade de recursos de hardware, a manutenção da consistência dos dados e o desempenho do sistema devem ser levados em consideração.

As operações que necessitam ítens de dados que não estão presentes naquele determinado instante, quando no caso dessas operações serem realizadas em modo de desconexão, podem ser bloqueadas ou gerar uma solicitação de busca dos dados que será feita em momento posterior.

As operações efetuadas localmente ao dispositivo devem ser armazenadas de alguma forma para posterior reconciliação com o BD consolidado. Estas ações podem ser guardadas em um arquivo de *log*, contendo os *timestamps* associados a estas mudanças com marcação através de *flags*. Tais ações podem ser armazenadas em um arquivo específico (*hoard*) ou ser associadas a própria base de dados.

Em um momento de reconexão, os dados replicado são reintegrados à base de dados consolidada. Esta operação também tem o nome de reconciliação, e pode envolver todos os registros, ou somente aqueles que foram colocados no *log* ou aqueles que foram marcados com *flags*. Este procedimento pode gerar alguns conflitos provenientes de atualizações concorrentes que podem ser resolvidas pela intervenção do usuário, pela utilização de *timestamps*, ou de forma

automática.

3.3.1 Modelos de Replicação

Nos últimos anos, foram propostos alguns modelos de replicação de dados, cujo objetivo principal é fazer com que as réplicas possam ser plenamente acessadas pelos dispositivo móveis, dando suporte ao compartilhamento de dados e sua conseqüente atualização. Como mencionado por (RATNER et al., 2001), alguns destes modelos de replicação são descritos nos ítens a seguir.

3.3.1.1 Replicação Cliente-Servidor

Neste modelo de replicação de dados, o processo de criação das réplicas pode partir tanto através de uma requisição do cliente, quanto do servidor, caso ele sabia quais dados o cliente deseja, já que em muitos casos não há espaço suficiente no dispositivo móvel para armazenar uma réplica integral do banco de dados. Entretanto, os dados a partir dos quais as réplicas serão construídas tem por origem um servidor, que normalmente é uma estação localizada na rede fixa, que possui alimentação contínua de energia e conectividade permanente à rede.

Este modelo tem por base um esquema de requisição-resposta. As requisições que partem dos dispositivos móveis são feitas somente às aplicações servidoras. Neste modelo não há comunicação direta entre os clientes. Caso isto seja necessário, o servidor atua como intermediador entre esta comunicação, como pode ser visto na Figura 11. Assim, o servidor é o nodo central de comunicação neste modelo, uma vez que possui a base de dados integral. Uma vez recebidas informações de atualizações pelos clientes, estas atualizações devem ser propagadas normalmente por *broadcast* aos demais clientes.

3.3.1.2 Replicação Ponto-a-Ponto

Uma característica intrínseca a computação móvel é a mobilidade proporcionada pelos dispositivos que fazem parte deste conjunto. Assim, não é possível prever sua exata localização em um determinado momento. De modo que é mais barato, rápido e eficiente comunicar-se

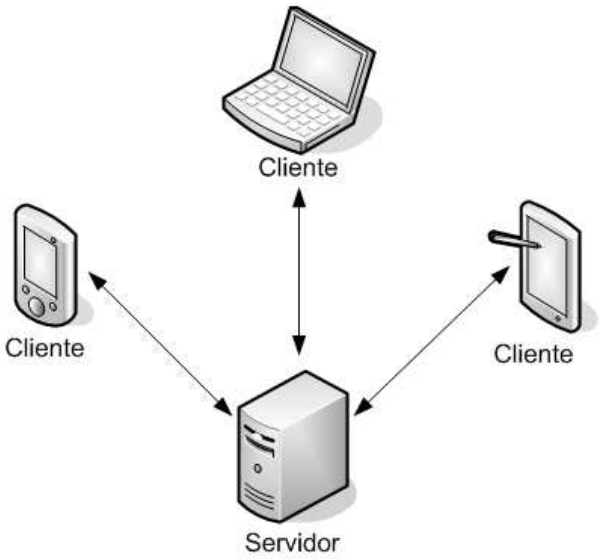

Figura 11: Replicação Cliente Servidor

localmente ao invés de efetuar uma conexão remota, os usuários de dispositivos móveis devem ter a habilidade de se comunicar com quem estiver mais perto.

(RATNER et al., 2001) afirma que é possível manter a consistência de forma efetiva mesmo que dois elementos não possam comunicar entre si, como em sistemas baseados no modelo de comunicação cliente-servidor. Contudo, uma sincronização local melhora a usabilidade e o nível de funcionalidades enquanto o custo inerente de sincronização diminui. Uma vez que os usuários da aplicação estão próximos, não há a necessidade de eventualmente fazer a reintegração através de uma conexão de longa distância, assim a sincronização é praticamente instantânea e envolve menor custo de transmissão (Figura 12).

Além disso, a replicação de dados de forma ponto-a-ponto diminui o tráfego na rede em comparação à replicação cliente-servidor devido ao fato de que, através desta última, para efetuar a sincronização entre dois clientes seria necessário passar obrigatoriamente por um servidor, envolvendo três elementos na comunicação ao invés de dois elementos apenas.

3.3.1.3 Replicação WARD

Segundo (RATNER; POPEK; REIHER, 1996), o modelo de replicação WARD (*Wide Area Replication Domain*), combina elementos clássicos dos modelos cliente-servidor e ponto-a-

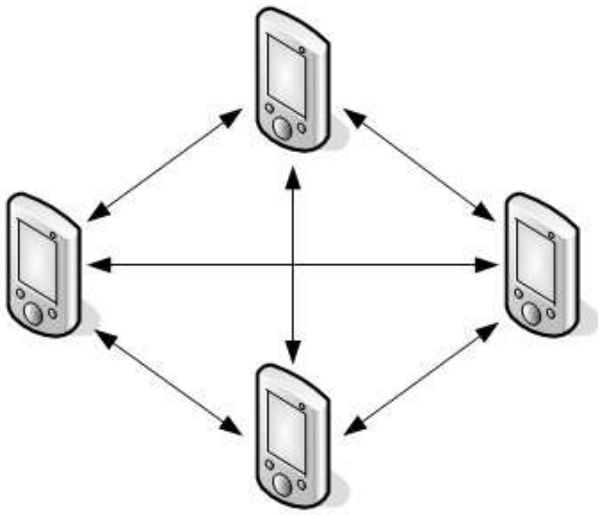

Figura 12: Replicação Ponto-a-Ponto

ponto, fornecendo uma solução escalável, com flexibilidade de replicação, possibilitando uma reconfiguração dinâmica do ponto de vista da topologia da rede. Além disso, uma WARD, é uma coleção de dispositivos que utilizam réplicas de dados e que estão próximos uns dos outros, como no caso de uma MANET.

De acordo com (RATNER et al., 2001), os WARDs são criados quando réplicas são adicionadas ao sistema. Cada nova réplica escolhe se vai se unir a um WARD já existente ou se vai formar um novo. Ratner et. al. acredita que é possível automatizar a designação de membros do WARD, mas isto é uma tarefa complexa e assim, esta designação foi feita de forma manual, através da intervenção de um administrador do sistema.

De forma que todos os membros do WARD são *peers*, o WARD designa um dos nodos como sendo o WARD Master, como um servidor numa arquitetura cliente-servidor tradicional, porém com algumas diferenças.

- De forma que todos os membros são *peers*, quaisquer dois membros podem efetuar a sincronização de seus dados entre si. Seja por acaso ou de forma proposital, os usuários irão freqüentemente encontrar outros usuários, e neste caso, a comunicação entre eles fica facilitada, barata e mais eficiente do que o acesso ao WARD Master;

- De forma que todos os membros são *peers*, qualquer membro do WARD pode atuar como

Master. Reeleição automática e reconfiguração do Master quando este falhar ou se tornar indisponível. A consistência não é diretamente afetada pela ausência temporária de um master, contudo, o sistema mantém um nível de consistência mais elevado se o WARD Master estiver sempre disponível e acessível;

- Não é necessário que o WARD Master armazene todos os dados dos objetos internos que o compõem, mas deve ser possível a ele que identifique o conjunto completo.

O WARD Master é o único canal de comunicação de um *peer* com outro WARD. Esta é uma forma pela qual o modelo atinge um nível de escalabilidade elevado, ou seja, pela limitação do montante de conhecimento armazenado em réplicas individuais.

Além disso, todos os WARD Masters pertencem a um nível elevado no WARD, formando uma hierarquia em dois níveis. O WARD Master atua de acordo com o comportamento do WARD, trazendo para ele novas atualizações, exportando outras para fora do WARD. A consistência é mantida por todas as réplicas pela comunicação direta de um Master com outros Masters e possibilitando a propagação independentemente em cada WARD, como mostrado na Figura 13.

Complementando, ainda pode haver a possibilidade de um dispositivo móvel atuar em dois ou mais WARDs, comunicando-se com os WARD Masters ou Members. Neste caso ele é chamado de Double Member, como também pode ser observado na Figura 13.

3.3.1.4 Replicação *Lazy* e *Eager*

De acordo com (WIESMANN et al., 2000), a replicação *lazy* (preguiçosa) é baseada em um modelo assíncrono de replicação e alguns SGBDs como DB2, Oracle e Bayou, implementam esta forma de replicação, quando alterações introduzidas por uma transação são propagadas para outros hosts, depois que esta transação foi efetivada(*committed*).

Este tipo de replicação resulta em um *overhead* mínimo, e neste caso inconsistências entre as réplicas são raras, mas podem aparecer. Contudo, isto não significa dizer que a consistência

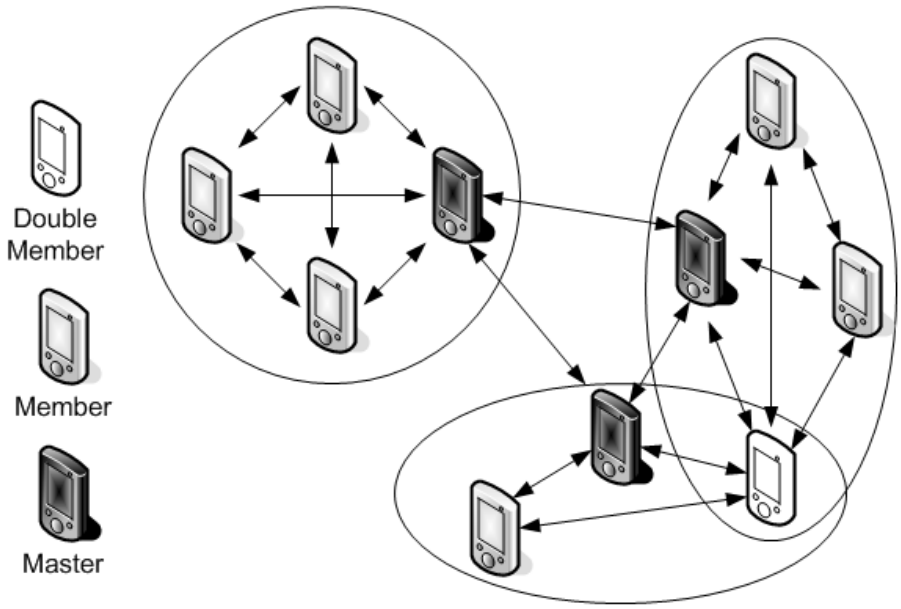

Figura 13: Arquitetura WARD

não é importante em bancos de dados.

É de conhecimento de usuários e desenvolvedores que as possíveis inconsistências geradas por técnicas de replicação preguiçosa podem ser difíceis de resolver. E é de conhecimento também, que estas inconsistências podem ser resolvidas usando modelos de replicação síncrona ou ansiosa, (também conhecida como *eager*), onde uma transação sincroniza com todas as réplicas antes de efetivá-la(*commit*). (GRAY et al., 1996) também ratifica que a replicação ansiosa mantém todas as réplicas igualmente sincronizadas, atualizando-as como uma transação indivisível(atômica).

Ainda segundo (GRAY et al., 1996), a replicação ansiosa não é uma opção a se considerar em um ambiente de computação móvel, no qual as desconexões podem ser freqüentes. Aplicações móveis necessitam da replicação preguiçosa que propaga as atualizações nas réplicas de forma assíncrona, depois que as transações foram efetivadas. Ainda assim, mesmo em estações fixas, a replicação preguiçosa é utilizada devido ao baixo *overhead* e tempo de resposta quando comparada à replicação ansiosa.

Complementando, a replicação ansiosa utiliza um esquema de bloqueio para garantir a serialização na execução de transações, enquanto que na maioria das implementações de replicação

preguiçosa, um esquema de controle de concorrência baseado em multi-versão para detectar possíveis comportamentos de não-serialização. Esquemas de isolamento multi-versão favorecem sempre o valor efetivado mais recente, baseado em *timestamping*.

A replicação ansiosa normalmente atrasa ou aborta transações não efetivadas quando uma violação de serialização é detectada. Por outro lado a replicação preguiçosa tem uma tarefa mais difícil quando isto ocorre, porque em algumas réplicas as transações podem já ter sido concluídas. Neste caso, não existe uma forma automática de efetuar o *rollback* das transações já efetivadas nas réplicas.

Infelizmente, o desenvolvimento de protocolos de replicação ansiosa não é trivial. Na prática, dadas às sérias limitações de técnicas de replicação tradicionais (*deadlocks*, *overhead*, escalabilidade e suposições não realistas), vários desenvolvedores de SGBDs não consideram a replicação ansiosa como praticável.

A maioria dos protocolos eficientes para replicação ansiosa é baseada em primitivas de comunicação de grupo e os resultados obtidos indicam que esta abordagem pode resolver a maioria dos problemas encontrados no desenvolvimento de protocolos de replicação ansiosa. Ainda em (WIESMANN et al., 2000), é mencionado que estas primitivas de comunicação de grupo podem ser embutidas em SGBDs e usadas como parte do gerenciamento de transações para assegurar a serialização na execução de tais transações em dados replicados.

As alterações nas réplicas podem ser controladas de duas formas: por um objeto master ou por grupo (sem objeto master), como pode ser observado na Figura 14:

- Grupo: alterações podem ser emanadas por qualquer nodo, de forma que a modificação de um ítem de dado pode acontecer em qualquer nodo. Entretanto, esta situação pode causar um número maior de conflitos

- Master: somente ele pode efetuar e propagar as alterações para as demais réplicas e todas as outras serão somente para leitura (*read-only*). Se outros nodos desejarem efetuar modificações, esta requisição deve ser enviada para o nodo master.

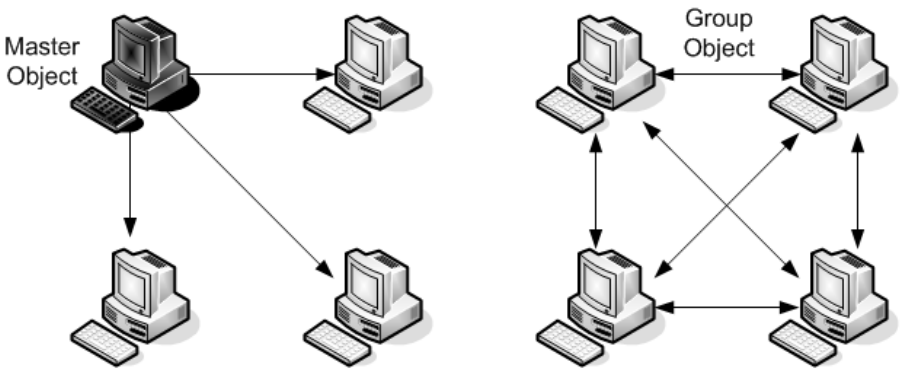

Figura 14: Controle de Alterações nas Réplicas

Propagação x Propriedade	Preguiçosa	Ansiosa
Grupo	*n* transações *n* donos de objeto	1 transação n donos de objeto
Master	*n* transações 1 dono de objeto	1 transação 1 dono de objeto
Duas Fases	n+1 transações, 1 dono de objeto atualizações locais, atualizações ansiosas	

Tabela 2: Taxonomia comparativa

Além disso, a Tabela 2, mostra um comparativo entre estratégias de replicação (*eager* ou *lazy*) e a estratégia de controle por propriedade (master ou grupo).

3.3.2 Protocolos de Replicação

Em ambientes de computação móvel, são comuns as desconexões das unidades móveis à rede fixa. Anteriormente já foram apresentados alguns motivos pelos quais isto pode ocorrer. Para evitar que problemas maiores ocorram, além da própria desconexão, é necessário que existam protocolos de replicação para assegurar ao usuário a manutenção dos dados de forma autônoma. Assim, as réplicas dos dados ficam armazenadas de modo persistente no dispositivo até o momento da reintegração dos dados. Este processo pode ser iniciado por qualquer parte envolvida no processo.

3.3.2.1 Replicação Otimista

A abordagem de replicação otimista envolve a possibilidade de acesso desconexo aos dados, sem a necessidade de reintegração imediata e instantânea após um acesso isolado. Assim, os dispositivos móveis podem efetuar a reintegração com a base consolidada, depois que efetuar um conjunto completo de operações sobre os dados locais. Esta abordagem pode ser comparada a alguns mecanismos de coerência de *cache* utilizados em sistemas de arquivos, tais como o CFS (*Coda File System*) (HELAL; KHUSHRAJ; ZHANG, 2002). Desta forma, elimina-se a necessidade de haver uma conexão permanente com o servidor, além de possibilitar um melhor uso da limitada energia disponível nos dispositivos móveis.

A replicação otimista permite que os dados apresentados aos usuários possam permanecer desatualizados por algum tempo, mas de modo controlado. Entretanto, havendo a possibilidade, é interessante, mas não obrigatório, que o usuário efetue a reintegração dos dados à base consolidada. Além disso, a reintegração pode ser feita em modo de *background*, possibilitando ao usuário que este continue trabalhando com os dados locais durante o processo de reintegração. Esta característica faz com que estes algoritmos sejam mais indicados quando se dispõe de poucos recursos de *hardware* e conectividade, como no caso de dispositivos móveis.

A conectividade dos dispositivos móveis pode variar com freqüência, o que leva a um estado temporário de possível inconsistência, além de incorrer em certos custos no que tange ao controle de concorrência, que segundo (AHUJA et al., 1999) podem ser tolerados por três motivos:

1. Em um pequeno período de tempo, a maioria dos ítens de dados tende a sofrer raras operações de escrita e como conseqüência, a geração de atualizações concorrentes sobre as réplicas é esporádica;

2. Muitas aplicações trabalham bem com informações desatualizadas, assim a propagação imediata de atualizações para as réplicas não é vital;

3. Grande parte dos acessos concorrentes a determinados ítens de dados podem ser executados em paralelo

Segundo (COGLIANESE, 2000), algumas decisões que devem ser tomadas no projeto de uma aplicação móvel que utiliza um protocolo otimista para replicação de dados incluem os seguintes ítens:

- **Arquitetura do Sistema e Comunicação**: o grau de comunicação permitido entre as diversas unidades que compõem o sistema deve ser observado. Pode ser empregado um modelo cliente-servidor ou um modelo *peer-to-peer*, ou um modelo híbrido (no qual os clientes podem se comunicar tanto com o servidor quanto entre eles mesmos;

- **Gerenciamento de *Cache***: é a forma que cada dispositivo representa e administra seu armazenamento persistente local. Perguntas comuns relacionadas a este item são: Que informações adicionais devem ser armazenadas com os dados para atingir um nível satisfatório de consistência? Como as atualizações de dados são representadas e gerenciadas?

- **Propagação e Reconciliação de Atualizações**: deve ser considerada o modo como as escritas no *cache* serão enviadas para as outras unidades pertencentes ao sistema. Em adição deve-se dedicar atenção especial quando atualizações conflitantes são reconciliadas: rejeitar uma das transações ou fazer um *merge* entre os dados?

- **Envolvimento da Aplicação**: Em alguns casos o sistema permite que a aplicação móvel tome parte no gerenciamento de *cache* e nas políticas de atualização. E a instância mais comum é na reconciliação de atualizações conflitantes.

De acordo com (SAITO; LEVY, 2000), a detecção e resolução de conflitos no mecanismo otimista pode ser aplicada não somente no momento de reconciliação dos dados, mas também em procedimentos isolados na unidade móvel. Desta forma, dá-se ao dispositivo uma autonomia maior na atualização de ítens de dados, mas por outro lado, também aumenta a complexidade do esquema, exigindo maior desempenho do dispositivo portátil.

3.3.2.2 Replicação Pessimista

Como relatado brevemente na seção anterior, a replicação pessimista se baseia na primitiva de transações atômicas atuando em todas as réplicas de forma síncrona. Desta forma o usuário tem a ilusão de haver uma única fonte de dados, altamente disponível. Contudo a comunicação entre as réplicas não pode apresentar falhas.

Conforme mencionado por (SAITO; LEVY, 2000), a expressão pessimista proíbe acesso a uma réplica a menos que o conteúdo seja atualizado. Embora técnicas como esta sejam essenciais em alguns tipos de aplicações, como para gerenciamento de transações bancárias, têm a desvantagem de exigir uma conexão permanente ao restante dos nodos que armazenam réplicas do banco de dados.

Além da conexão ininterrupta entre os nodos, há um *overhead* maior neste algoritmo (e conseqüente perda de desempenho), devido ao fato da sincronização demandar maiores recursos de comunicação entre eles.

3.3.2.3 Considerações sobre os Protocolos de Replicação Otimista e Pessimista

A replicação de dados em múltiplos *sites* em sistemas de bancos de dados distribuídos envolve um aumento de desempenho e disponibilidade. Entretanto, dependendo da localização geográfica do usuário ou do tipo de usuário envolvido no processo, a utilização de mecanismos específicos é fundamental para se obter um nível elevado de consistência no processamento de transações em bancos de dados distribuídos. Em (SAITO; SHAPIRO, 2005), estão descritos alguns aspectos aos quais se deve dar uma atenção particular, como segue:

- **Mobilidade**: de forma que no modelo otimista as transações efetuadas em um banco de dados localizado em um dispositivo móvel são realizadas localmente em seu armazenamento persistente, não é obrigatório que o usuário permaneça sob cobertura de um ponto de acesso *wireless* à rede fixa. Já no modelo pessimista, esta característica torna impraticável uma operação realizada em dados locais, de forma que é necessária uma conexão

permanente para realizar a sincronização imediata dos dados com todas as réplicas;

- **Comunicação**: a autonomia de um dispositivo móvel é melhorada pelos protocolos de replicação otimistas, de forma que a comunicação entre as réplicas não necessita ser imediata. Transações podem ser submetidas a uma réplica local, sem haver a obrigatoriedade de propagá-la para as demais réplicas instantaneamente, reduzindo significativamente o *overhead* associado aos protocolos de replicação pessimistas;

- **Consistência**: no modelo otimista, a existência de transações concorrentes sobre ítens de dados localizados em diversas réplicas pode resultar na invalidação de transações submetidas por outros usuários. Além disso, operações de consulta podem retornar dados desatualizados, uma vez que estes só serão atualizados no momento de uma reintegração. Por outro lado, o modelo pessimista possui um mecanismo de controle de concorrência que consegue garantir a integridade e a atomicidade na execução de transações concorrentes;

- **Recursos**: como já visto, os recursos disponíveis em um ambiente de computação móvel são limitados quando comparados a estações fixas. Na proposta pessimista, todas as atualizações são consolidadas diretamente a todas as réplicas e no próprio banco de dados consolidado, através de troca de mensagens. No caso da computação móvel, esta troca de mensagens ocorre por meio de uma rede sem fio. Devido à limitação de recursos e largura de banda nestas redes, a atualização por meio de troca de mensagens acaba se tornando um fator limitador, porque deve suportar uma quantidade grande de requisições de usuários ocorrendo paralelamente. Da mesma forma que numa rede fixa, todas as requisições são atendidas por um nó central, e este deve possuir recursos de memória, rede, processamento e armazenamento suficiente para atender todas estas requisições, além de coordenar a propagação das atualizações para as demais réplicas. Já a proposta otimista dá ênfase na alta disponibilidade e desempenho no acesso aos dados, mantendo parte dos dados necessários ao usuário no seu próprio dispositivo, conforme seu perfil. Desta forma, a demanda por recursos nas requisições pode ser suprida localmente, diminuindo

a necessidade de recursos adicionais para suporte a comunicação;

- **Disponibilidade**: de acordo com o volume de requisições submetidas pelos usuários à base fixa, pode ocorrer que o número de transações pendentes seja maior do que a capacidade de atendimento pelo servidor. Isto acarreta uma perda de desempenho e aumento na latência destas transações para os clientes, prejudicando de certa forma a disponibilidade de informações ao cliente. Entretanto, no modelo otimista as transações são restritas à disponibilidade de recursos locais, não sendo afetada pela performance do servidor, permitindo que estas transações sejam executadas mais rapidamente, porém com menor grau de consistência;

- **Tolerância a Falhas**: protocolos otimistas de replicação de dados fornecem resultados satisfatórios em redes lentas ou incertas, de forma que conseguem propagar os dados para o banco consolidado sem efetuar o bloqueio de outras réplicas disponíveis na rede. Esta característica é essencial em ambientes móveis, nos quais as unidades móveis comunicam-se entre si com uma freqüência não tão grande. Porém, no protocolo pessimista, as réplicas têm seus acessos impedidos até o término da execução de uma transação e conseqüentemente o armazenamento do conteúdo mais atual.

3.4 Particionamento

Segundo (OZSU; VALDURIEZ, 1999), os registros de um banco de dados podem ser divididos em um ou mais pedaços denominados fragmentos. Estes fragmentos podem ser mantidos por diferentes computadores, mesmo geograficamente distribuídos. Entretanto, a localização das partições deve ser transparente para o usuário, de forma que o usuário não necessite conhecer onde estão localizados os fragmentos para poder submeter uma transação.

De acordo com (BARROSO; DEAN; HOLZLE, 2003), enquanto que a replicação fornece a característica de alta-disponibilidade (os dados se encontram em mais de um servidor), a fragmentação ou particionamento do banco de dados lhe garante uma melhor performance, visto que a consulta pode ser distribuída e processada em máquinas diferentes, cada uma acessando

um fragmento, enquanto que numa arquitetura não fragmentada, uma única máquina teria que realizar a consulta. Depois de obtidas todas as respostas de cada fragmento, os mesmos são unidos para devolver ao usuário uma resposta única.

A fragmentação pode ocorrer de três formas: horizontal, vertical ou mista.

- Fragmentação Horizontal: caracteriza-se pela divisão do banco de dados por linhas ou registros completos;
- Fragmentação Vertical: a divisão do banco de dados é feita utilizando-se de colunas, atributos ou campos, sendo que a chave primária deve estar presente em todos os fragmentos[1];
- Fragmentação Mista: ocorre tanto no sentido vertical quando no sentido horizontal.

Em bancos de dados móveis, dificilmente acontece a replicação completa de um banco de dados, por limitações já mencionadas acima, como espaço em disco e uso da rede. Conforme (TERRY et al., 1998), um dispositivo móvel contém normalmente réplicas parciais ou fragmentos a fim de suprir a necessidade de informação pelo usuário até o instante da próxima requisição. Este procedimento, como já citado anteriormente, denomina-se *caching*, que possui a finalidade de prover um acesso mais eficiente aos dados, evitando o uso da rede a cada solicitação. Por outro lado, as informações contidas no *cache* podem sofrer mudanças e estas devem necessariamente serem refletidas aos outros repositórios de dados no momento da consolidação.

3.5 Disseminação

De acordo com (BARBARA, 1999), a disseminação de dados pode ser definida como a entrega de dados de um conjunto de produtores para um conjunto de clientes, ou por analogia, consumidores. Devido à assimetria na comunicação entre clientes e servidores, a disseminação de dados é amplamente utilizada visto que a largura de banda no sentido servidor-cliente (*down-*

[1] desta forma há a duplicação de alguns atributos

link) é maior do que no sentido cliente-servidor (*uplink*). A disseminação pode ser classificada de acordo com o conteúdo, o modo de *broadcast* e o tipo.

Segundo uma taxonomia proposta por (CAI; TAN; OOI, 1997), de acordo com o conteúdo, a disseminação de dados pode ser classificada de dois modos:

- **Orientados a registro**: os registros completos são propagados para os clientes. Também são chamadas de *Propagações de Alterações*;
- **Orientados a ID**: neste caso, apenas os identificadores dos registros (chaves primárias) são propagados para os clientes. Esta técnica também é conhecida por *Invalidação de Alteração*.

Conforme (HOU et al., 2001), de acordo com o modo de *broadcast* utilizado, a disseminação de dados pode ser dividida de duas formas:

- **Assíncrono**: neste modo o servidor envia um *broadcast* contendo uma notificação de invalidação, informando aos receptores que este item de dado foi alterado. Contudo, outras informações além do identificador do registro podem ser enviadas, como por exemplo o *timestamp* da alteração mais recente daquele ítem de dado. Também pode ser enviado um relatório contendo todos os ítens de dados alterados recentemente. Assim, um cliente que porventura tenha permanecido desconectado da rede por um certo período pode aguardar o recebimento do próximo relatório de invalidação para saber se os ítens de dados que estão em seu *cache* local foram alterados ou não. Contudo, neste modo não há como saber quando será enviado o próximo relatório. A maior desvantagem desta abordagem é o tempo imprevisível de espera para as mensagens que dependem de atividades de alteração nos dados;
- **Síncrono**: neste modo, a disseminação de dados ocorre em *broadcasts* periódicos, dentro de um espaço de tempo pré-definido pela aplicação servidora. Alguma latência pode aparecer entre o momento da atualização real do ítem de dado e o momento do recebimento

da notificação. Depois de recebida a notificação, o cliente precisa verificar se o seu *cache* local continua válido. Caso isto não ocorra, o cliente deve solicitar ao servidor que este atualize seu *cache*.

Em (CHUNG; CHO, 1998), os autores afirmam que o modo síncrono é vantajoso quando um cliente é reconectado após algum período de desconexão. No entanto, o modo assíncrono não provê qualquer garantia de latência para seus clientes. Se o cliente faz uma consulta por um ítem de dado após o período de desconexão, este também precisa esperar a próxima notificação ou submeter uma consulta para o servidor para saber se os seus dados locais são válidos. No modo assíncrono porém, existe uma garantia de latência devido à característica periódica desta abordagem.

Com relação ao tipo de disseminação de dados, existem três classificações, como informado em (HARA, 2002):

- ***Pull-Based***: Nesta abordagem, conforme mencionado por (ITANI; DIAB; ARTAIL, 2005a), os clientes solicitam explicitamente as informações desejadas através do envio de requisições ao servidor, o qual responde de acordo com a solicitação. Desta forma, o servidor entrega os dados separadamente para cada cliente, conforme pode ser observado na Figura 15.

- ***Push-Based***: Nesta abordagem, as informações são enviadas para os clientes sem esperar que estes lhe solicitem, como mostrado na Figura 16. Contudo, estes sistemas tem o problema de decidir quais são os dados importantes que devem ser enviados para os clientes (BARBARA, 1999). O servidor também precisa determinar se os dados serão enviados periodicamente ou não. Envios periódicos têm a vantagem de permitir que os clientes sejam desconectados por algum tempo, e caso estes recebam um *broadcast* depois de reconectados, não percam seus ítens de dados. Por outro lado, o envio não periódico utiliza melhor a largura de banda disponível na rede. De acordo com (HARA, 2002), esta técnica consegue enviar mais dados por unidade de tempo que o *pull-based*, porque em geral a

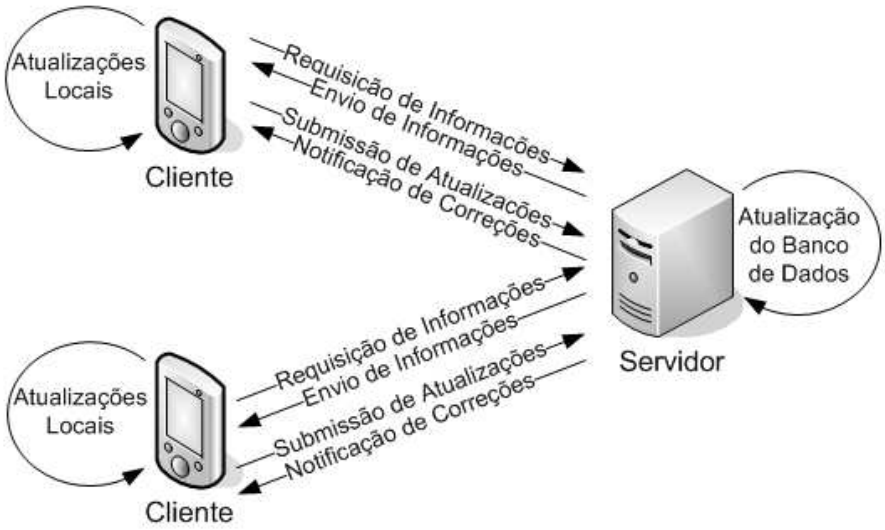

Figura 15: Disseminação *Pull-Based*

velocidade de *downlink* é maior que o *uplink*, além de poder enviar de forma coletiva os dados, já que na técnica *pull-based* o envio dos dados aos clientes é feita de forma individual.

- ***Hybrid-Mode***: Combinação das duas abordagens anteriores (*pull-based* e *push-based*), também conhecida como IPP (*Interleaved Puss and Pull*). De acordo com (BARBARA, 1999), esta abordagem pode ser proveitosa tanto para o servidor quanto para os clientes. Nesta abordagem, tanto as informações podem ser requisitadas de forma explícita aos dispositivos que fazem parte do sistema, quanto o envio das informações pode ser feito em modo de *broadcast*, sem que o cliente faça uma requisição específica, considerando uma periodicidade ou não (síncrona ou assíncrona) neste envio (Figura 17).

3.6 Reconciliação

Como já relatado anteriormente, os dispositivos móveis possuem uma conexão intermitente com a rede sem fio. Estas unidades armazenam dados em seus *caches* locais, onde podem efetuar alterações. Estas alterações fazem com que se torne necessária uma conexão periódica para reintegração dos dados alterados, para que outros usuário que porventura possam utilizá-los tenham na medida do possível, os dados mais atualizados. Vez por outra, podem

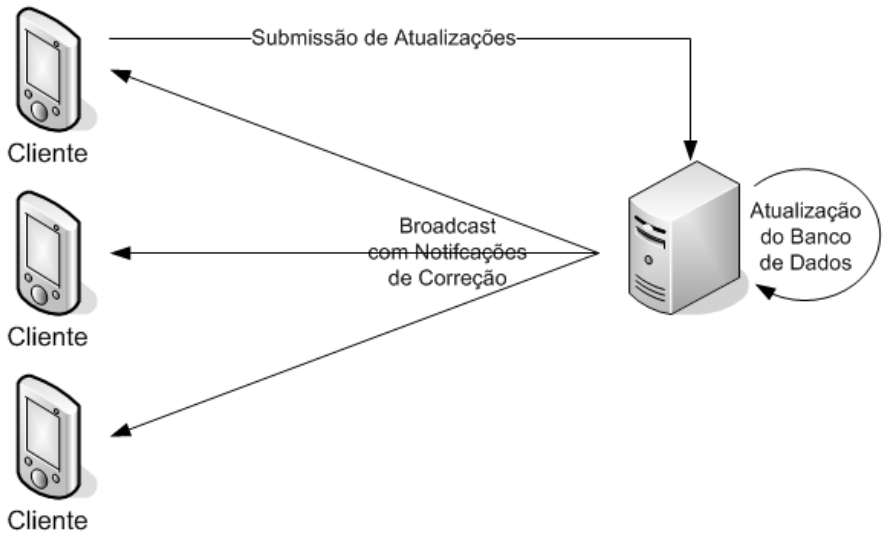

Figura 16: Disseminação *Push-Based*

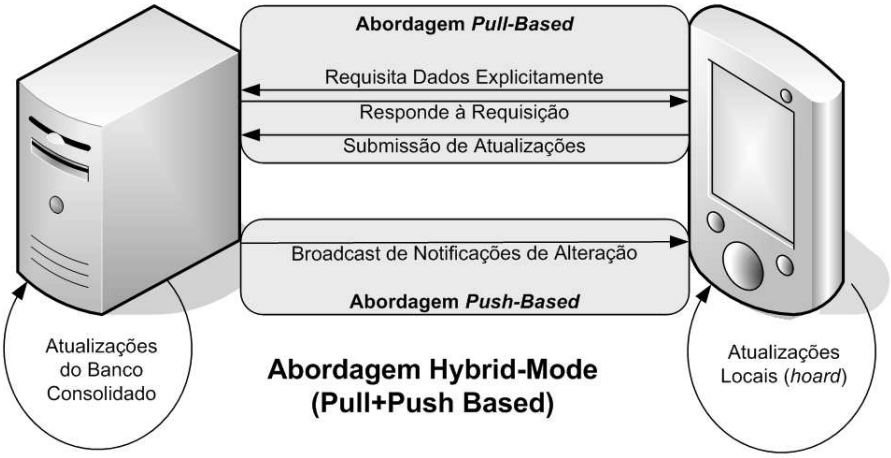

Figura 17: Disseminação *Hybrid-Mode (Pull+Push Based*

ocorrer problemas de execução de transações concorrentes gerando conflitos, os quais devem ser solucionados. Esta operação de reintegração também é conhecida como reconciliação ou sincronização.

A reconciliação em ambientes móveis pode ser definida como o ato de estabelecer equivalência entre duas coleções de dados, entre cliente e servidor, após a ocorrência de alteração nos registros armazenados de forma persistente (PHATAK; BADRINATH, 1999). Após este procedimento de sincronização de dados, cada ítem de dado em uma coleção é mapeado por um ítem de dado em outra coleção, sendo que seus dados são equivalentes, no entanto não necessitam ser obrigatoriamente idênticos.

As transações submetidas a uma fonte de dados localizada em um dispositivo móvel são replicadas, porém de forma assíncrona, devido à possível ausência de conexão à rede sem fio estruturada.

Quanto aos modelos utilizados para propagação de alterações feitas nos dados locais de um dispositivo móvel, existem três modelos amplamente aceitos:

- ***Session-Based***: Neste modelo de sincronização a replicação de dados é efetuada por meio de uma comunicação direta entre a base móvel e a base consolidada. O dispositivo móvel inicia então a comunicação de sincronização pelo envio de uma lista completa das modificações feitas naquela base desde a última sincronização. O servidor envia de volta então as modificações relevantes. O elemento móvel inicia então o processo de incorporação das mudanças e envia uma confirmação de sucesso da operação juntamente com uma tabela que mapeia o conteúdo que foi atualizado (*mapping table*). A sessão de comunicação neste contexto consiste de:

  - Validação de segurança em cada sessão;
  - A comunicação de dados propriamente dita;
  - Responder as requisições de um cliente remoto;
  - Distribuição eletrônica de software sob direção do servidor central;

  - Monitoramento de recursos e diagnósticos;

  Neste contexto, o processo de *staging* considera as atividades a serem realizadas numa sessão de forma enfileirada. Assim, este processo pode ser considerado como uma quantidade de trabalho a ser executado em momento oportuno. Os elementos podem acumular sessões de trabalho, analisar a sua execução e controlar o tempo, tipo de conexão ou disponibilidade de recursos.

- ***Message-Based***: A troca de dados entre as bases também podem ser realizadas através da troca de mensagens, que normalmente são arquivos armazenados em algum diretório particular ou mensagens de correio eletrônico com formatos específicos. Segundo (PARK; YEOM, 2000), um agente de mensagens vinculado a cada elemento da rede pode enviar mensagens informando as mudanças recentes em sua base e receber mensagens de um ou mais agentes, modificando seus dados de acordo com seu conteúdo.

  Em (PARK; WOO; YEOM, 2002), é mencionado que este modelo de sincronização permite replicação/reconciliação de bancos que não estão diretamente conectados. Cada mensagem carrega informações de controle (como seu endereço) para que nenhuma conexão seja necessária entre as aplicações que se comunicam remotamente.

- ***Connection-Based***: Neste modelo, os dispositivos móveis dependem da existência de uma área de cobertura para ser estabelecida uma conexão com o provedor de serviços. Em um ambiente móvel, as desconexões e re-conexões ocorrem freqüentemente devido à instabilidade da comunicação pela rede sem fio.

  Para amenizar esta situação, (FUKUSHIMA; TAKAHASHI; NARAZAKI, 2001) propôs um método que estima o tempo de conexão e velocidade de movimentação do cliente usando o envio de sinais provenientes de uma estação base. Através desta alternativa, é possível estabilizar a conexão entre os dispositivos móveis, melhorando a transferência completa de dados em operações de reconciliação.

3.7 Mecanismo de *Timestamp*

Em sistemas de gerenciamento de banco de dados, existem dois métodos para implementação do controle de concorrência: o protocolo de bloqueio e o protocolo de *timestamp*. Em uma abordagem pessimista de computação móvel, não é possível utilizar o protocolo de bloqueio devido ao fato de que a conexão do dispositivo móvel com o servidor de dados não é permanente. É possível sua utilização somente se considerarmos um ambiente otimista, no qual os dispositivos estão sempre conectados ao servidor através de uma conexão de rede sem fio.

Diferentemente dos algoritmos baseados em bloqueio, o protocolo de timestamp não tenta manter a serialização no controle de transações por exclusão mútua. Neste caso, a implementação do sistema de banco de dados estabelece a ordem de serialização através do *timestamp*, que é um identificador simples, que contém a data e a hora que determinado dado foi atualizado. Como exemplo, podemos ter "2007-01-12 23:59:59". A informação de milisegundos pode não estar incluída, dependendo da implementação do sistema gerenciador de banco de dados. Assim, através do *timestamp*, é possível determinar se um dado entrante é mais recente ou mais antigo do que o dado já existente, caso exista.

Existem várias opções para atribuir *timestamps* para transações ou dados: uma delas é utilizar um gerador de timestamp global na área de cobertura da rede. Contudo, a sincronização e manutenção de geradores de *timestamps* globais é um desafio para a área de sistemas distribuídos. Por outro lado, é possível simplesmente assumir que os relógios dos dispositivos estão sincronizados e cada dispositivo pode gerar seu próprio *timestamp*. Para garantir uma geração de *timestamps* mais sincronizada na rede, o NTP (*Network Time Protocol*) pode ser utilizado. Este protocolo está disponível em redes TCP/IP para sincronização de relógios de geradores de *timestamps* para atingir um grau mais elevado de consistência na atualização de dados.

4 *Modelo Proposto*

Antes de apresentar os motivos que serviram como alicerce para proposição do modelo, é necessário mostrar alguns trabalhos correlatos que possuem características semelhantes às que serão apresentadas a seguir.

4.1 Trabalhos Correlatos

Carla Berkenbrock (BERKENBROCK; DANTAS, 2005), em seu trabalho de investigação e análise de desempenho de estratégias de coerência de cache em ambientes cliente-servidor móveis, realizou seu experimento utilizando uma aplicação similar que trata de coerência e consistência de cache local usando mecanismo de *timestamping*. Contudo a aplicação desenvolvida em seu trabalho não realizava atualização de dados. A aplicação somente lia e inseria dados novos, e os dados inseridos não eram tratados pelo mecanismo de consistência.

Cunha (CUNHA; DANTAS, 2004), em um estudo de caso experimental sobre replicação e reconciliação em ambientes de redes sem fio, desenvolveu um mecanismo para replicação e reconciliação de dados entre clientes móveis e servidor. Contudo, a aplicação desenvolvida no experimento não permitia o compartilhamento de dados entre os próprios clientes.

XMIDDLE (ZACHARIADIS et al., 2002), consiste em um *middleware* para compartilhamento de informações que possibilita a manipulação de dados em modo de desconexão e utiliza o formato XML (eXtensible Markup Language) para envio e recebimento de dados. Contudo, pela característica da linguagem XML, há um *overhead* causado pelas *tags* da linguagem, que acaba por consumir mais largura de banda da rede sem fio e mais espaço disponível no dispositivo

móvel para armazenamento persistente dos dados.

Roam (RATNER; REIHER; POPEK, 1999), é um sistema de replicação de dados projetado para computação móvel, porém, utiliza uma abordagem otimista, ou seja, presume que todas as réplicas estejam conectadas à rede no momento da reconciliação. Também utiliza um modelo de comunicação ponto-a-ponto, permitindo que duas réplicas façam a sincronização entre si. Também utiliza o conceito de *Wards*, que são coleções de nodos próximos, nos quais é permitida a sincronização sem intermediários.

Trabalhos recentes, tais como em (YIN; CAO, 2006) e (CAO et al., 2007) denominam este compartilhamento de dados utilizando a arquitetura cliente-servidor tradicional, bem como através de comunicação ponto-a-ponto entre as unidades móveis, de *Cooperative Caching*. No primeiro trabalho, foram propostos três esquemas de consistência que os autores chamaram de *CachePath*, *CacheData* e *HybridCache*, que é uma combinação das vantagens dos dois anteriores, além de utilizar políticas de substituição de *cache*. No segundo trabalho foi proposto um modelo genérico e híbrido de consistência que utiliza as principais características de esquemas de consistência de *cache*, baseado em técnicas de invalidação de ítens, que foi chamado de RPCC (*Relay Peer based Cache Consistency*).

4.2 Motivação

O problema abordado neste trabalho de pesquisa teve origem na FAMEG (Faculdade Metropolitana de Guaramirim), na qual o autor é docente dos cursos de Tecnologia em Sistemas de Informação e em Redes de Computadores. Esta instituição possui um ERP (*Enterprise Resource Planning*) Educacional denominado SIGAF (Sistema Integrado de Gestão Acadêmica e Financeira), (KNAESEL, 2007). Este ERP foi totalmente desenvolvido visando acesso remoto através da Web, utilizando como plataforma softwares livres tais como a linguagem PHP (Hypertext Pre-Processor), e o Sistema de Gerenciador de Banco de Dados MySQL. Através do "Módulo Professor", é possível que os docentes atualizem as notas e freqüência dos discentes, bem como dos conteúdos ministrados. Contudo, diversos professores do curso possuem

seus computadores portáteis e alguns deles PDAs, e infelizmente, a rede sem fio existente na instituição não cobre todos os ambientes e salas de aula, impossibilitando o acesso imediato ao sistema para lançamento das notas, freqüência e conteúdos ministrados. Desta forma, pensou-se em desenvolver uma aplicação que possibilitasse ao docente efetuar estes lançamentos, mesmo estando desconectados da rede, e que pudesse atualizá-los na base de dados centralizada assim que entrasse na área de cobertura da rede. Também há casos onde dois docentes lecionam a mesma disciplina para a mesma turma, em horários diferentes, como é o caso da disciplina de algoritmos, onde um professor leciona conteúdos teóricos em sala de aula e outro efetua a parte prática da disciplina em laboratório. Assim, também há necessidade de compartilhar as informações entre dois (ou mais) professores. Isto posto, pensou-se em elaborar um modelo computacional que pudesse atender esta necessidade da academia.

4.3 Arquitetura do Modelo Proposto

O objetivo principal deste trabalho de pesquisa, é o de ter uma aplicação leve que possa compartilhar e atualizar dados de forma consistente entre dispositivos móveis e servidores estacionários. Complementando, é importante considerar a troca de dados entre os dispositivos móveis em redes estruturadas e também em redes ad-hoc. Faz parte do objetivo deste trabalho a utilização de formatos simples para troca e armazenamento de dados para consumir menos largura de banda da rede e menos espaço de armazenamento no cliente móvel, visto que a maioria deles possui recursos extremamente limitados.

Um certo número de algoritmos foram propostos para arquiteturas cliente-servidor convencionais (BJORNSSON; SHRIRA, 2002). Do mesmo modo, existe uma tendência da utilização destas arquiteturas cliente-servidor em ambientes de computação móvel (HOLLIDAY; AGRAWAL; ABBADI, 2002). No protótipo desenvolvido, foi utilizada uma arquitetura cliente-servidor para a comunicação entre o cliente móvel e o servidor estacionário na rede cabeada através das estações de controle; entre os dispositivos móveis também foi utilizada uma abordagem *peer-to-peer*, na qual um dispositivo móvel atua como cliente e outro atua como servidor em redes

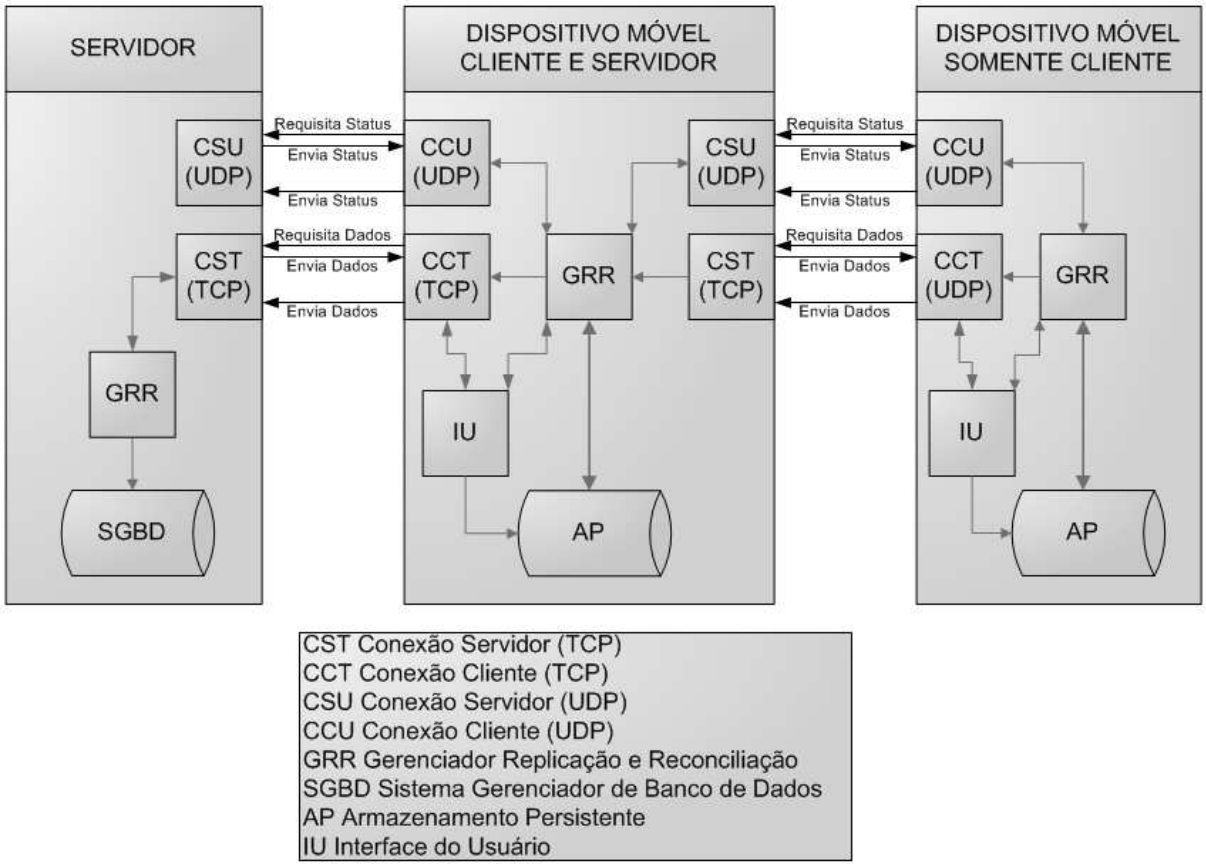

Figura 18: Arquitetura do Modelo Proposto

sem fio com estações de controle ou redes sem fio ad-hoc. O mecanismo de replicação utilizado foi baseado no modelo de replicação preguiçosa (*lazy*), visto que alguns autores como (JIMENEZ-PERIS et al., 2001; WIESMANN et al., 2000; GRAY et al., 1996) consideram este como sendo o modelo mais indicado para replicação de dados em ambientes móveis. O modo de disseminação de dados escolhido foi o *pull-based*, no qual o servidor só envia a informação quando solicitado, de modo que os dispositivos móveis podem desconectar-se da rede freqüentemente. A Figura 18 representa a arquitetura do modelo proposto.

Conforme observado na Figura 18, a arquitetura do modelo proposto é formada pelos seguintes componentes:

- **Conexão Servidor TCP(CST)**: Composto por um *TCP (Transmission Control Protocol) server socket*, é responsável pelas atividades de receber e atender requisições dos usuários (envio de resposta), bem como receber dados provenientes de outros usuários.

Este componente está disponível tanto no servidor estacionário, quanto nos dispositivos móveis;

- **Conexão Cliente TCP(CCT)**: Composto por um TCP *client socket*, é responsável pelo envio das requisições de dados, recebimento das respectivas respostas, e também pelo envio de dados para outro dispositivo;

- **Conexão Servidor UDP(CSU)**: Composto por um *UDP (User Datagram Protocol) server socket*, é responsável pelas atividades de receber e atender requisições de status do dispositivo, bem como receber os status de outros dispositivos. Este componente está disponível tanto no servidor estacionário, quanto nos dispositivos móveis;

- **Conexão Cliente UDP(CCU)**: Composto por um UDP *client socket*, é responsável pelo envio das requisições de status dos dispositivos, recebimento das respectivas respostas de status, e também pelo envio de status para outros dispositivos;

- **Gerenciador de Replicação e Reconciliação(GRR)**: Este é um dos componentes principais do protótipo. Depois de recebida uma requisição ou um dado, este é o componente que atua como intermediário entre as conexões cliente/servidor e o SGBD ou o armazenamento persistente de dados do dispositivo móvel.

  Quando este componente recebe uma requisição de dados (*Request Data*), ele a processa verificando se o dado requisitado está disponível na base de dados (SGBD ou armazenamento persistente). Caso o dado requisitado esteja disponível, este é devolvido. Ao final é enviada uma mensagem de fim de transmissão e é solicitado o fechamento da conexão.

  Este componente também recebe dados propriamente ditos pela resposta de uma requisição ou de envio de dados. Quando os dados são recebidos, é verificado se cada item de dado já existe na base através de sua chave primária. Caso não exista, o registro é inserido. Caso não exista e seja igual ao existente, nenhuma operação é realizada. Caso não exista e seja diferente, seus *timestamps* são comparados e se o registro entrante for mais recente, uma operação de atualização é realizada. Complementando, este componente está disponível tanto no servidor estacionário, quanto nos dispositivos móveis;

- **Sistema Gerenciador de Banco de Dados(SGBD)**: Localizado no servidor estacionário, é o responsável por manter a base de dados consolidada, a qual, além de ser acessada pelos clientes móveis, também é acessada por outras aplicações legadas (no cenário descrito, é acessado pelo ERP educacional (SIGAF) em uso pela FAMEG;

- **Armazenamento Persistente(AP)**: Composto pelo armazenamento persistente de dados, está localizado nos dispositivos móveis, onde estão as réplicas dos dados;

- **Interface com Usuário(IU)**: É através desta que o usuário pode efetuar modificações nos dados existentes no armazenamento persistente, bem como fazer as solicitações de busca e envio de dados de/para o servidor estacionário ou outros dispositivos móveis que estão em sua área de abrangência, através de um ponto de acesso ou de forma ad-hoc.

Os diagramas UML (*Unified Modeling Language*) de atividade ilustram melhor o funcionamento do modelo proposto, do ponto de vista do cliente (Figura 19), e do ponto de vista do servidor (Figura 20).

Na Figura 21, é possível observar o Diagrama de Atividades correspondente à atividade de detectar quais são os clientes e servidores disponíveis através da rede sem fio.

4.4 Arquitetura do Protótipo

O protótipo representa um sistema de informação no qual professores podem armazenar as presenças e ausências de seus alunos em suas turmas. Um determinado professor pode compartilhar uma mesma disciplina com outro professor. Por exemplo: uma disciplina pode ter dois professores, um lecionando a parte teórica e outro a parte prática.

O Diagramas de Base de Dados utilizado no protótipo da aplicação servidora pode ser visto na Figura 22. Para a aplicação móvel este modelo foi modificado para reduzir o número de tabelas e conseqüentemente o número de pesquisas em chaves primárias e estrangeiras. O Diagrama da Base de Dados da aplicação móvel pode ser visto na Figura 23.

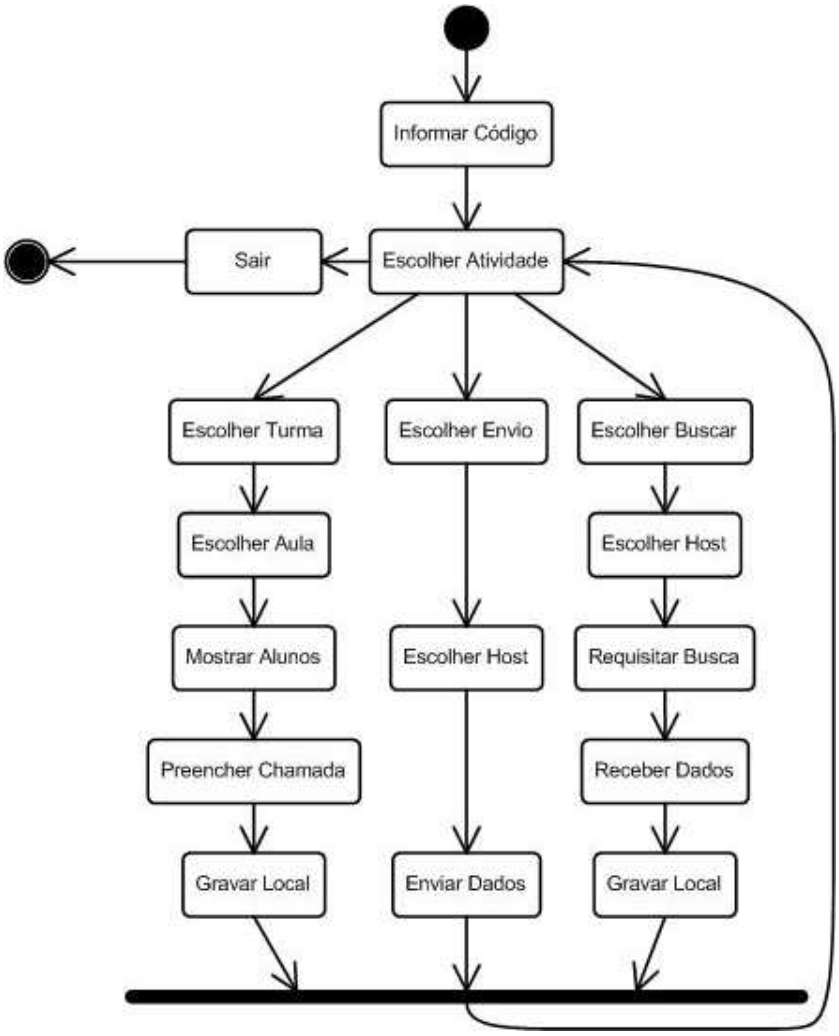

Figura 19: Diagrama de Atividades - Ponto de Vista da Função Cliente

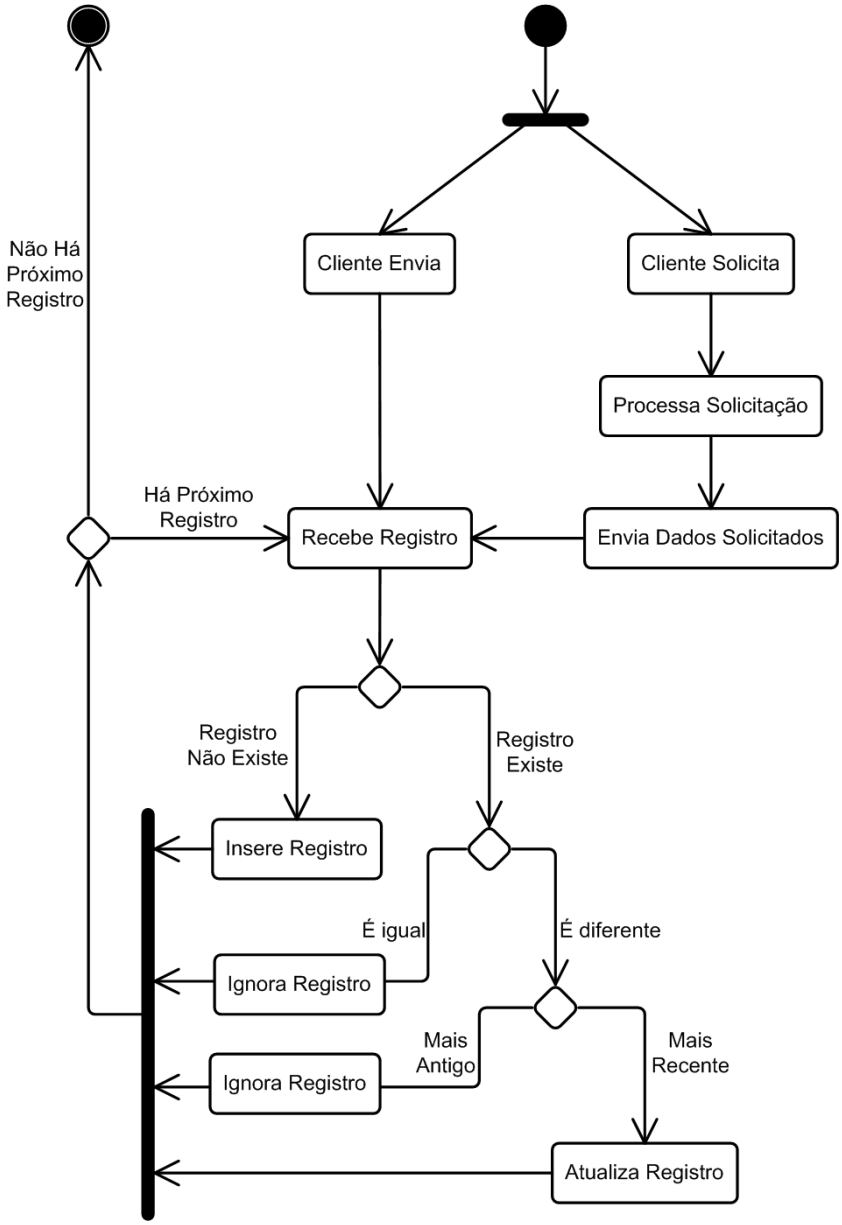

Figura 20: Diagrama de Atividades - Ponto de Vista da Função de Servidor

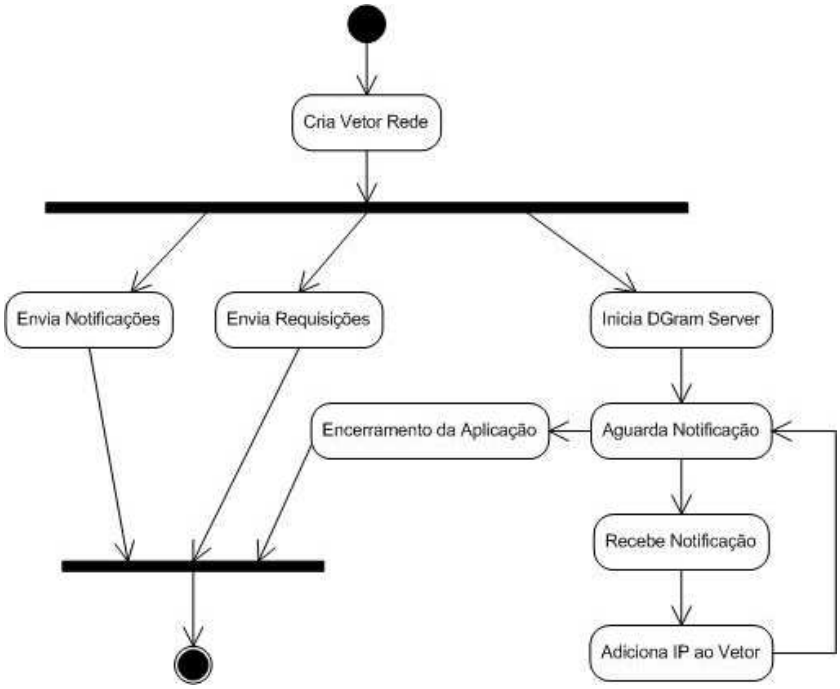

Figura 21: Diagrama de Atividades - Detecção dos Hosts da Rede Sem Fio

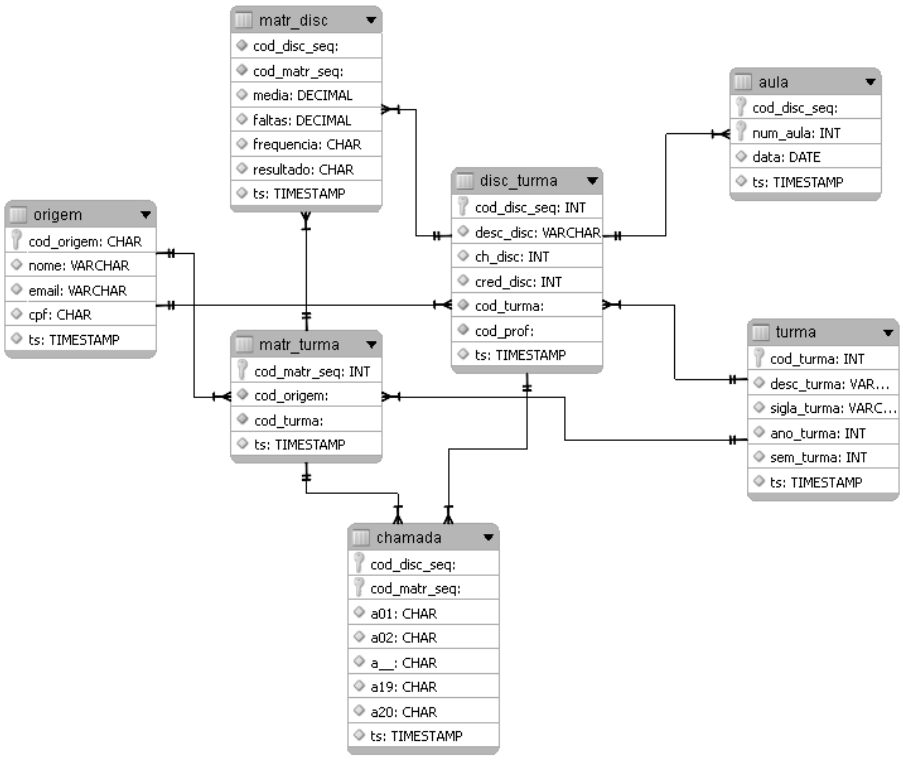

Figura 22: Diagrama da Base de Dados da Aplicação Servidora

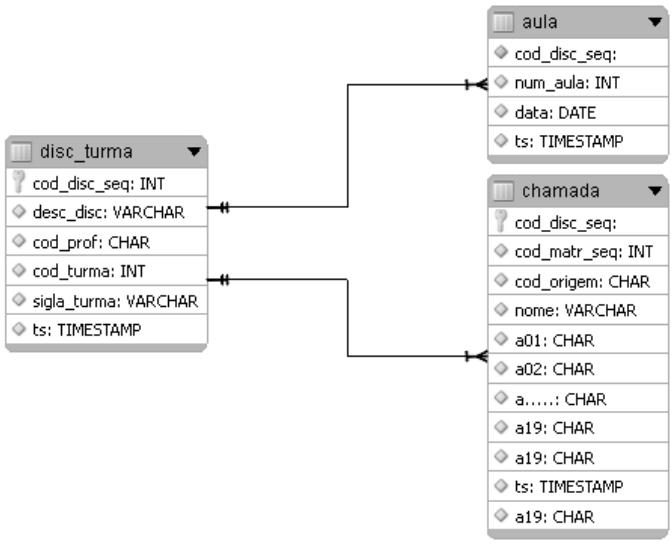

Figura 23: Diagrama da Base de Dados da Aplicação Móvel

Para a comunicação dos dados entre clientes e servidor, bem como entre os clientes propriamente ditos, foram utilizados *socket streams* TCP (Figura 25) e envio de datagramas UDP. Os datagramas (pacotes UDP) foram utilizados no protótipo para descobrir quais os dispositivos estão na área de cobertura do dispositivo, de forma *unicast*, ou seja, é enviada uma mensagem para cada endereço IP presente na configuração. Pelo fato de uma conexão UDP não ser orientada a conexão, foi esta a conexão utilizada para descoberta dos dispositivos ativos na rede, pois esta não fica bloqueada aguardando uma resposta do destinatário.

As conexões TCP foram utilizadas para efetuar o recebimento e envio de dados, pois este tipo de conexão possui garantia de entrega dos pacotes. É importante observar na figura que num primeiro instante o socket servidor é iniciado e posteriormente um cliente pode conectar-se a ele. Um cliente envia uma requisição para o servidor através de uma primitiva *write()*, este lê (*read()*) a requisição e a processa. Em seguida ele devolve a resposta (*write()*) e o cliente a recebe (*read()*). E este procedimento continua até que o cliente solicite o fechamento da conexão (*close()*).

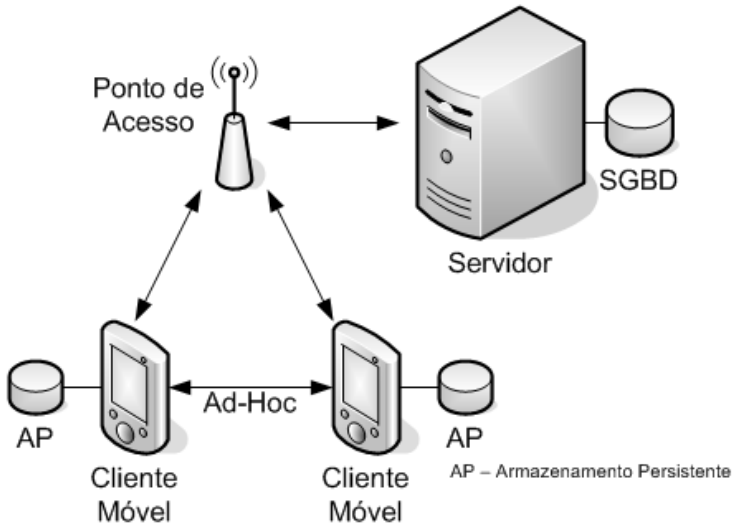

Figura 24: Ambiente Experimental

4.5 Ambiente Experimental

O ambiente experimental utilizado é composto por dois clientes móveis e um servidor localizado na rede fixa. Os dispositivos móveis requisitam e enviam informações de/para um servidor estacionário que mantém um SGBD contendo os dados de uma aplicação já existente através de uma rede sem fio com pontos de acesso. Neste ambiente também é possível fazer a comunicação entre os clientes móveis utilizando redes sem fio com pontos de acesso ou *ad-hoc*, conforme apresentado na Figura 24.

Para implementar as aplicações que executam no servidor estacionário e no cliente móvel (que também pode atuar como servidor móvel), foi utilizada a linguagem de programação Java com a API J2SE (para a aplicação que roda no servidor estacionário) e a API J2ME (para a aplicação móvel). A versão da máquina virtual Java utilizada para o desenvolvimento do protótipo foi a 1.5.0_07, também conhecida como versão 5.0 *update* 7.

A configuração de hardware e software utilizada no ambiente experimental foi:

- servidor estacionário: computador desktop com processador AMD Duron 2400 com 512Mb de memória RAM, rodando a aplicação servidor acessando um banco de dados MySQL através de uma conexão JDBC com o "Java MySQL Connector", disponível no

website do próprio MySQL;

- dois dispositivos móveis modelo Palm Tungsten C com processador Intel de 400MHz, 64Mb de memória interna, sistema operacional PalmOS versão 5.2.1, com interface Wi-Fi (802.11b), e a máquina virtual Java utilizada foi a IBM WebSphere Everyplace Micro-Environment 5.7;

- para o desenvolvimento de ambas aplicações foi utilizado o Eclipse SDK com o plug-in EclipseME e o Sun Java Wireless Toolkit.

O SGBD MySQL foi escolhido por ser *open-source*, além de prover um tempo de resposta rápido às consultas e prover as funcionalidades necessárias para implementação do protótipo como integridade referencial (BAZGHANDI, 2006; GIACOMO, 2005). A linguagem Java foi escolhida por executar bem em dispositivos móveis com pouca memória e capacidade de processamento, vasta documentação pode ser encontrada na Internet, além de possuir as funcionalidades necessárias para armazenamento persistente e de rede, tais como *socket streams* e datagramas (KIMM; SHIN; SHIM, 2005).

4.6 Funcionamento do Protótipo baseado no Modelo

Esta seção descreve detalhadamente o funcionamento do modelo proposto, com referência à implementação do protótipo.

4.6.1 Procedimentos Iniciais

Inicialmente, a aplicação no dispositivo móvel solicita ao cliente que ele informe seu código[1]. Em seguida são enviados datagramas notificando os demais agentes da rede que este está ativo. Esta mensagem é uma *string* no formato CSV (*Comma Separated Values*) e é constituída pela palavra "sim", seguida da palavra "client"(pois este é um cliente móvel) e finalizada com o

[1]Na aplicação implementada não foram consideradas questões de segurança, desta forma não é necessário que o docente informe sua senha.

endereço IP local, como por exemplo: "sim;client;172.16.1.21". Os endereços IP destinatários da mensagem são gerados dentro de uma estrutura de repetição, que no caso desta aplicação, são enviados datagramas para os endereços 172.16.1.15 a 172.16.1.25.

Por outro lado, a aplicação fica no outro dispositivo móvel escutando por datagramas. Se a aplicação recebe o datagrama com a mensagem acima, ela armazena o endereço IP do datagrama recebido em um vetor na memória volátil do dispositivo. Os endereços IP recebidos não são armazenados de forma persistente no dispositivo porque eles podem mudar freqüentemente. Esta notificação foi feita para informar outros dispositivos móveis na área de abrangência da rede sem fio estruturada ou ad-hoc, que aquele dispositivo está disponível, no caso de ele ser ligado ou entrar na área de abrangência da rede sem fio após o servidor e/ou outros dispositivos móveis.

Além desta notificação, que informa outros dispositivos de sua disponibilidade, são enviadas requisições para saber quais são os outros hosts disponíveis na rede. Esta requisição também segue o formato CSV e é composta pelos elementos: "alive", endereço IP remoto (gerado da mesma forma que no procedimento anterior) e o endereço IP local, como por exemplo "alive;172.16.1.15;172.16.1.21". Semelhante ao processo anterior, o processo que recebe datagramas, analisa o conteúdo da mensagem recebida e caso esta mensagem tenha sido esta requisição, gera nova mensagem para ser devolvida ao elemento requisitante. Esta mensagem segue o mesmo formato citado no parágrafo anterior.

A aplicação desenvolvida que é executada no servidor, também escuta por datagramas e devolve as respectivas respostas aos clientes, com a diferença que o servidor não envia notificações nem requisições, porém este recebe requisições e envia respostas. É necessário ressaltar também, que na resposta enviada ao requisitante, a palavra "client", é substituída por "server", informando que este dispositivo é assim, um servidor.

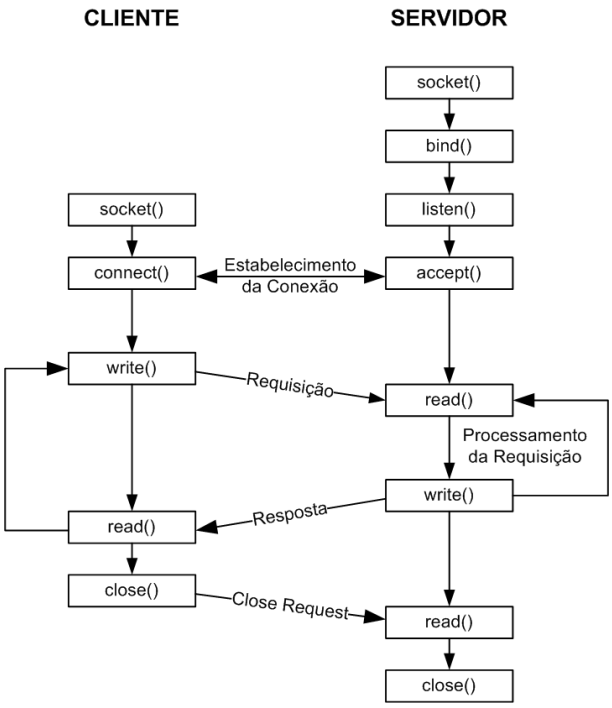

Figura 25: Função *Socket*

4.6.2 Aquisição e Envio de Dados a Outros Dispositivos

Depois deste processo de verificação de quais dispositivos estão ativos, um *menu* é apresentado ao usuário com as opções de "Buscar Dados"ou "Enviar Dados". Estes dois procedimentos iniciais podem ser vistos na Figura 26. Caso existam turmas/disciplinas cadastradas e relacionadas com o código do professor informado no início, estas também serão apresentadas (o procedimento que é executado quando uma turma/disciplina é selecionada será descrito mais adiante).

Se a primeira opção (buscar dados) foi selecionada, é apresentado um formulário com os *hosts* que foram descobertos no processo anterior. Depois de escolhido o *host*, a aplicação envia novamente um datagrama ao *host* selecionado requisitando a disponibilidade do mesmo. A mensagem contida no datagrama é idêntica à mensagem mostrada no primeiro parágrafo desta seção, ou seja, "alive;remIpAddr;localIpAddr". Se o *host* selecionado receber este datagrama, ele responde com a mensagem "sim;client|server;localIpAddr". Caso ele tenha recebido esta resposta, então a aplicação inicia uma conexão tipo *socket stream* TCP com o *host* selecionado, enviando a requisição dos dados. Esta requisição contém a mensagem:

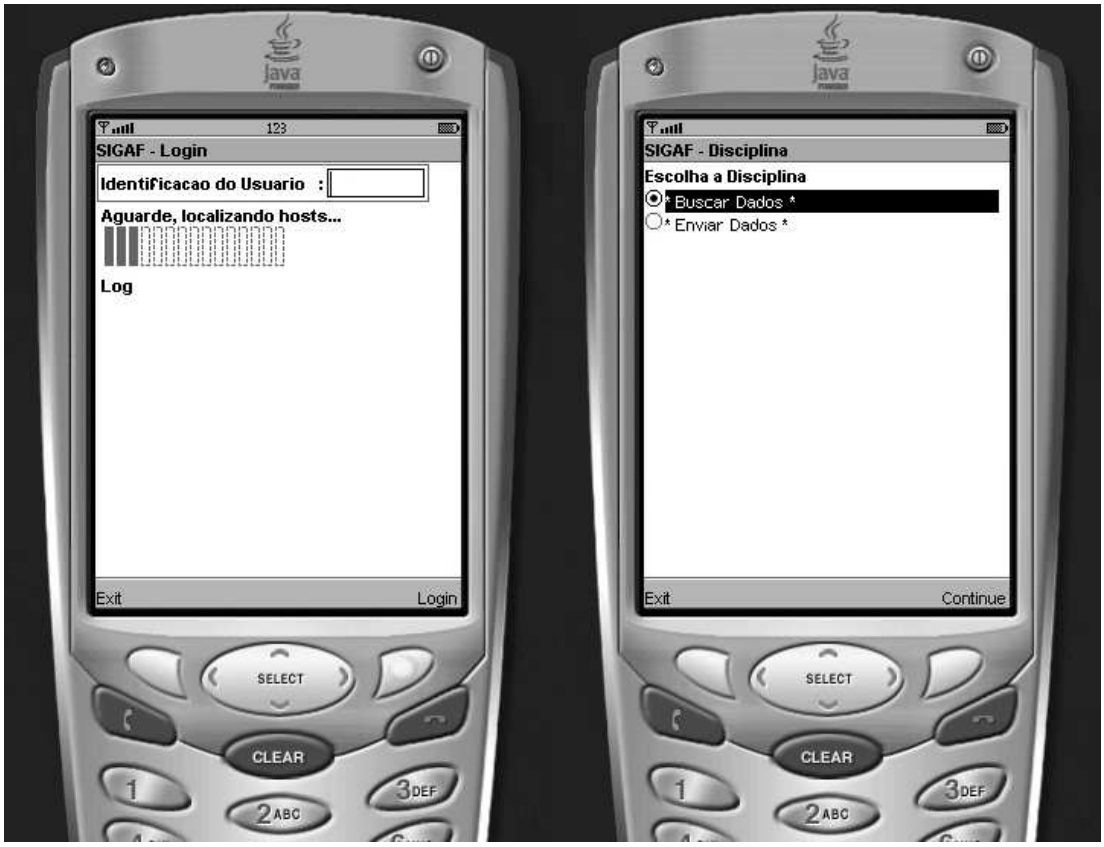

Figura 26: Telas de Login e de Opções Busca/Envio de Dados

"busca_dados;tabela;cod_prof", sendo que tabela é o nome da tabela/*Record Store* de onde os dados serão extraídos e "cod_prof"é o código do professor, informado na primeira tela.

O dispositivo que atua como servidor envia o dado requisitado através desta conexão. Ao mesmo tempo, não apenas os dispositivos móveis ficam escutando por datagramas e requisições, mas também o servidor estacionário, porque os dados podem ser recebidos tanto de um dispositivo móvel como de um servidor. Este procedimento pode ser observado na Figura 27.

Quando um determinado item de dado chega ao dispositivo móvel (pela conexão com o servidor estacionário ou pela conexão com outro dispositivo móvel, se ele possui o dado requisitado), a aplicação inicia então o processo de armazenamento persistente dos dados recebidos usando a Classe *Record Management System* (RMS) da Java API para dispositivos móveis. Cada registro recebido pela conexão *socket* é verificado se ele já existe no armazenamento persistente de dados. Caso o registro não exista, a aplicação então insere o registro no armazenamento persistente. Se o registro já existe, a aplicação então compara os dois registros. Se os

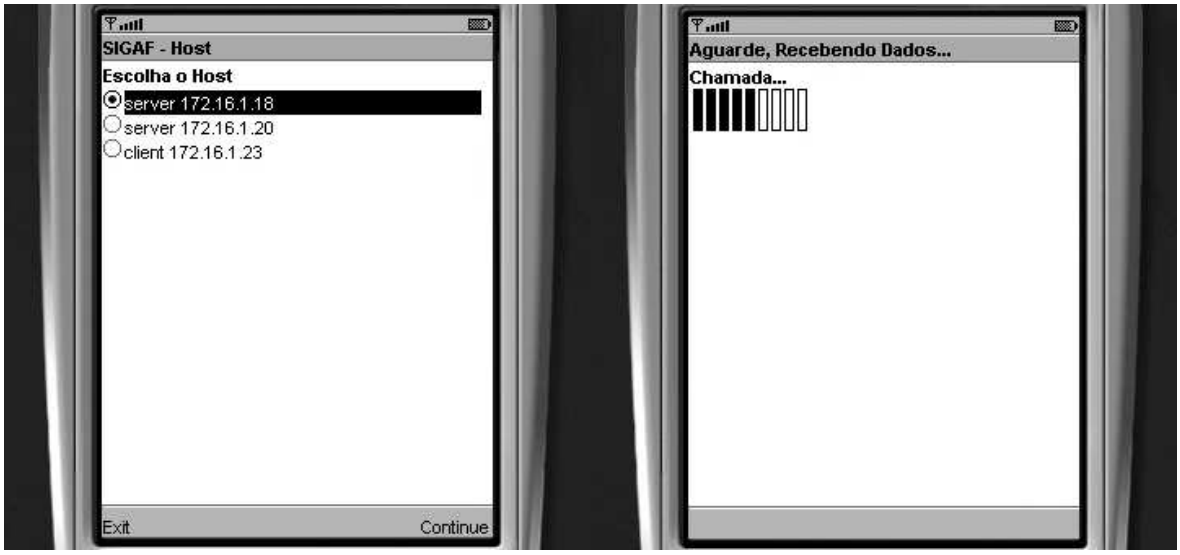

Figura 27: Telas de Seleção do Host e do Progresso da Busca dos Dados

registros forem iguais, a atualização não é necessária. Caso contrário, ou seja, se os registros forem diferentes, a aplicação compara então o *timestamp* do registro recebido pela conexão e do registro já localizado. Se o *timestamp* do registro entrante for mais recente, então a aplicação faz a atualização dos dados do registro com o novo *timestamp*. Este processo pode ser visto através da Figura 20.

Também durante o processo de recebimento de dados, é necessário mencionar que para o usuário, é interessante que os registros estejam ordenados. Por exemplo, quando o professor marca as presenças e ausências dos alunos seus nomes devem estar alfabeticamente ordenados. Isto foi feito na aplicação usando uma simples e recursiva implementação do algoritmo de ordenação *quicksort*. O mesmo processo de comparação dos registros entrantes e seus respectivos *timestamps* também ocorre quando um dado é recebido do servidor estacionário.

Por outro lado, existe uma opção, acima mencionada, que envia dados armazenados de forma persistente no dispositivo móvel. A mensagem que contém a requisição de envio, é: "manda_dados;tabela". Este procedimento envia todos os dados armazenados no dispositivo móvel. O envio dos dados foi implementado desta forma, para aumentar a garantia de que os dados serão consolidados na base de dados do servidor. Exemplificando, o cliente, pode enviar os dados para qualquer dispositivo que esteja na sua área de abrangência. Se por algum

motivo alheio à sua vontade os dados ou o próprio dispositivo forem extraviados, haverá uma cópia atualizada dos dados que ele possui em outro dispositivo, e assim que este efetuar o envio para o servidor, seus dados estão de certa forma, protegidos. Complementando, o dispositivo que está recebendo os dados, efetua a mesma comparação dos registros entrantes com os já armazenados para inserção e atualização, quando for o caso (Figura 20).

O envio e recebimento de datagramas, assim como as conexões tipo *socket* para envio e recebimento de dados podem ser feitas em redes sem fio com estações de controle (*access points*) ou em redes sem fio ad-hoc. Isto é importante devido a característica de desconexões freqüentes dos dispositivos móveis por causa da área de cobertura das estações de controle, ruído ou falta de energia nas baterias. Já, o processo de comunicação entre o dispositivo móvel e o servidor estacionário obrigatoriamente deve passar por uma estação de controle porque a conectividade com o servidor é cabeada e a estação de controle está em sua grande parte, conectada a uma rede cabeada.

4.6.3 Atualização dos Dados Locais

A aplicação implementada também possui uma interface para o usuário para interagir com os dados armazenados localmente. Nesta aplicação, não é possível realizar a inserção de novos registros, porque eles vêm de outra aplicação, que o Departamento Acadêmico da Universidade mantém. Assim, depois do recebimento dos dados, o professor pode escolher a disciplina que ele leciona. Isto feito, é apresentado um novo formulário no qual ele escolhe qual a aula que ele deseja através de um número inteiro seqüencial como por exemplo: 1 (primeira aula), 2 (segunda aula), e assim por diante. Caso uma destas aulas já tenha sido preenchida, a data na qual foi lecionada aquela aula será mostrada. Estes procedimentos podem ser vistos na Figura 28.

Em momento posterior à escolha da aula pelo usuário, um novo formulário é apresentado no qual ele pode informar a data na qual aquela aula aconteceu, e abaixo, a lista com o nome dos alunos onde é possível marcar as ausências dos alunos. Tanto no SIGAF, quanto no cliente

Figura 28: Telas de Seleção da Turma e da Aula

móvel, a opção pelo preenchimento das ausências, ao invés das presenças, foi escolhido porque normalmente o número de alunos presentes é maior que o número de alunos ausentes, visto que segundo a LDB (Lei de Diretrizes e Bases da Educação), nº 9.394, de 20 de dezembro de 1996, em seu artigo 24, inciso VI, determina que é exigida freqüência de no mínimo 75% do total de horas letivas para aprovação.

Depois de preenchido o formulário, o cliente pode salvar os dados localmente. O processo que salva localmente os dados busca os registros do formulário e localiza o registro correspondente no armazenamento persistente do dispositivo. Depois de localizado, ele é atualizado com o novo *timestamp* gerado localmente.

4.6.4 Envio de Dados para Outros Dispositivos

Após este processo de atualização local dos dados, o usuário pode enviar os novos dados para um servidor estacionário ou para outro cliente móvel que compartilha os dados com ele, ou para outro cliente por motivos de alta disponibilidade e segurança, porque um problema pode acontecer com o dispositivo móvel e acabar perdendo todos os dados. Através deste processo, ele garante que os dados que ele atualizou estarão seguros no servidor ou em outro dispositivo móvel. Os procedimento de preenchimento dos dados no dispositivo móvel e posterior envio

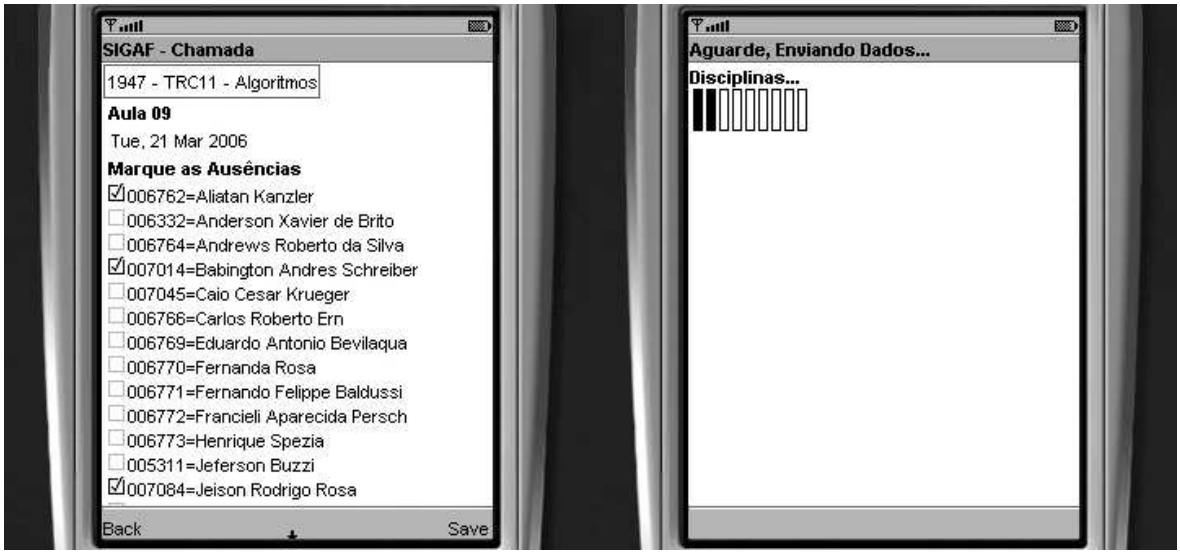

Figura 29: Telas de Preenchimento da Chamada por parte do Docente e Progresso de Envio dos Dados para o Servidor ou outro Dispositivo Móvel

para um determinado host podem ser vistos na Figura 29

É também importante mencionar que os dados que são transferidos e armazenados localmente no dispositivo móvel estão no formato CSV, para reduzir a quantidade de espaço necessário no dispositivo para armazenamento dos dados, para reduzir volume de dados trafegados na rede sem fio e para reduzir o tempo de transferência, localização e armazenamento dos dados.

4.6.5 Relato dos Experimentos Realizados

Inicialmente, a parte da base de dados do Sistema Acadêmico necessária para o experimento foi implantada no SGBD do Servidor e a aplicação servidora do protótipo foi inicializada.

Em seguida um dispositivo móvel foi inicializado com a aplicação móvel, um código de usuário foi informado e a opção buscar dados foi selecionada a partir do servidor detectado no procedimento de busca dos nós da rede. Os dados foram recebidos e comparados com o que havia na base de dados consolidada e nenhuma inconsistência foi detectada.

Logo após este procedimento, algumas alterações locais ao dispositivo móvel foram realizadas e devidamente gravadas através do armazenamento persistente. Os dados foram enviados ao servidor e comparados para verificar se havia alguma inconsistência. Isto feito, foi possível

concluir que o procedimento cliente-servidor estava funcionando como esperado, atualizando corretamente as marcas de tempo (*timestamp*) e o conteúdo dos registros.

Após isto, o outro dispositivo móvel foi inicializado e a busca de dados foi realizada selecionando como fonte dos dados a primeira estação móvel. Alterações locais foram realizadas, gravadas e enviadas de volta ao dispositivo móvel de origem dos dados; este, por sua vez, realizou o envio para a base consolidada no servidor e o conteúdo dos registros foi verificado um a um, obtendo desta forma, a igualdade entre todos os registros.

Contudo, como a abordagem de replicação utilizada foi a *pull-based*, é necessário que os clientes efetuem o recebimento dos dados necessários para manipulação a partir do servidor estacionário ou de outra estação móvel que esteja compartilhando dados com esta, de forma explícita, antes de efetuar as alterações locais; ou ainda, que a estação móvel que efetuou a alteração envie estes dados o mais rápido possível para o servidor ou para o dispositivo móvel com o qual esteja compartilhando dados, caso contrário, dados alterados por outro usuário podem ser sobrepostos, como se estes fossem os mais recentes. Se isto não for realizado, não há como garantir um nível máximo de consistência nos dados.

Este fato também ocorre pelo fato de a base de dados do cliente móvel estar desnormalizada, ou seja, um registro contém informações de freqüência não de apenas uma aula, mas sim de todas as aulas do período letivo em questão. Normalizando a base de dados, o nível de consistência também pode ser aumentado.

Ao se adotar uma abordagem híbrida (*pull*+*push*-based), este problema também pode ser parcialmente resolvido pois o envio dos dados para servidor e clientes móveis é feito de forma implícita, periodicamente ou não, sem conhecimento e intervenção do usuário.

Possíveis inconsistências também poderiam ser tratadas por um esquema de multi-versões do banco de dados na aplicação servidora. Caso haja mais versões de uma mesma informação, deve-se escolher uma dentre elas ou realizar um *merge* entre as versões encontradas para obter uma única versão.

Observados estes pontos, ou seja, a necessidade de envio dos dados após a alteração ou

recebimento dos dados antes da realização de atualizações, testes reais foram realizados em uma disciplina que é compartilhada pelo autor e outro usuário. Paralelamente, foram feitas anotações utilizando meios não computacionais para gestão destes dados (diário de classe). Durante o período letivo, os dados do servidor e dos dispositivos móveis foram confrontados por diversas vezes e ao final do período, o nível de consistência atingido foi considerado satisfatório pelos usuários e pela instituição em questão.

5 *Considerações Finais*

5.1 Conclusões

A partir das limitações impostas pelos dispositivos móveis, tais como largura de banda da rede sem fio, desconexões freqüentes e espaço reduzido para armazenamento dos dados, constantemente surgem idéias para suprir estas limitações e armazenar os dados localmente nestes elementos é uma solução comumente adotada.

Existem diversos trabalhos correlatos sobre o compartilhamento de dados em ambientes de computação móvel, mas nenhum deles atendia exatamente a idéia que havia sido proposta. A maioria dos trabalhos correlatos analisados tratavam de aplicações com dados somente para leitura ou da arquitetura cliente-servidor tradicional e não permitiam a comunicação entre os próprios clientes. Contudo, trabalhos recentes de *Cooperative Caching* tais como (YIN; CAO, 2006; CAO et al., 2007) fornecem esta funcionalidade, apesar de não terem sido implementados e experimentados na prática.

Uma das principais preocupações com este trabalho, era a de que ele não fosse apenas um trabalho teórico, mas que principalmente, pudesse gerar uma implementação e um experimento prático, que auxiliaria tecnologicamente os docentes em suas tarefas burocráticas, na qual o preenchimento do diário de classe é uma delas.

Nesta dissertação, foi proposto e implementado um mecanismo para compartilhamento consistente de dados entre cliente móvel e servidor estacionário, bem como entre os próprios clientes móveis. Também foi possível compartilhar os dados em redes sem fio ad-hoc, e não somente em redes sem fio com estações de controle. O dispositivo móvel, nesta abordagem, pode atuar

como cliente somente ou também como servidor para outros dispositivos móveis. Também foram utilizados formatos simples e leves para envio, recebimento e armazenamento dos dados.

Foram feitos inúmeros testes de preenchimento, gravação, envio e recepção dos dados e este apresentou funcionamento satisfatório. Outro diferencial da presente dissertação é que a mesma não foi apenas uma simulação, ela foi uma implementação com funcionalidades reais em um ambiente experimental também real, considerando o número de professores, turmas, disciplinas e funcionou de forma satisfatória.

Apesar da utilização de formatos leves para armazenamento, envio e recebimento dos dados, não foram realizadas medições de tempo nos procedimentos citados. Assim não foi possível concluir se o protótipo atende a requisitos de desempenho, até porque estes não foram especificados por não serem objetivo principal desta pesquisa. Entretanto, levando em conta a quantidade de dados manipulados em cada tarefa, pode-se estimar que este tempo não deve ser grande.

5.2 Limitações e Problemas Encontrados

Como limitações deste trabalho, podemos destacar a não preocupação com a segurança no sentido de que usuários não autorizados tenham acesso aos dados. Também pode-se colocar aqui, que poucos trabalhos deste gênero geram soluções livres, que possam facilmente ser utilizadas por outros acadêmicos e/ou desenvolvedores.

Também podemos aqui destacar que o não conhecimento prévio e limitações impostas por algumas ferramentas, tecnologias ou linguagens de programação, dentre as quais a principal é a ausência de métodos conhecidos de classes da plataforma J2SE, não presentes na plataforma J2ME, fizeram com que vários destes métodos tivessem sido reimplementados.

Outro item que é importante aqui colocar, é que o tratamento dos registros no formato CSV, no que tange a chaves primárias e acesso aos campos de cada item de dados foi feito de forma manual, ou seja, não havia uma camada que fizesse com que o trabalho de acesso e manipulação

dos dados fosse feito de forma transparente.

Uma limitação que merece ser notada, é a questão da não implementação no protótipo, de esquemas de sincronização ou atualização dos relógios dos dispositivos móveis. É através do *timestamp* que as informações contidas no servidor e nos clientes móveis são validadas e assim, estes precisam estar atualizados. No caso específico da aplicação desenvolvida para simulação do modelo, como a freqüência de atualização dos dados é relativamente baixa (em média duas vezes por semana), esta sincronização não é tão importante.

Apesar do modelo e protótipo ter atendido a necessidade da instituição durante o período de experimentação, foi detectado que por limitação do modelo, podem acontecer problemas de sobreposição de dados, visto que somente o *timestamp* é considerado para identificar qual item de dado seria o correto, além da necessidade do usuário em enviar os dados após a alteração e recebimento antes desta. Podem ocorrer casos em que dois usuários alterem o mesmo ítem de dado, mas que o que efetuou a alteração por último, tenha alterado de forma incorreta, e assim, invalidando a alteração efetuada com *timestamp* mais antigo (que seria a correta), gerando conflitos.

5.3 Trabalhos Futuros

Da presente pesquisa, pode-se visualizar algumas direções para trabalhos futuros, quais são:

- considerar o uso de uma solução livre ou comercial de banco de dados para dispositivos móveis, que possua suporte à linguagem SQL, evitando de que os registros tenham que ser tratados de forma manual;
- utilização de datagramas em *broadcast* para compartilhamento dos dados numa abordagem não-orientada a conexão, pois durante a transmissão dos dados podem ocorrer desconexões. Ou ainda, considerar a questão de desconexões durante a recepção ou envio dos dados;

- ampliar o número de funcionalidades disponíveis na aplicação disponibilizada aos professores, proporcionando aos mesmos o lançamento não somente do controle de freqüência dos alunos, mas também do lançamento das notas e dos conteúdos ministrados em cada aula;
- utilização de um algoritmo de compressão/compactação para reduzir ainda mais o volume de dados trafegados pela rede, bem como para reduzir o espaço de armazenamento persistente necessário nos dispositivos móveis;
- inclusão de um esquema de multi-versão nos ítens de dados, para possível resolução de conflitos;
- implementação no protótipo de um mecanismo de sincronização ou atualização de relógios;
- efetuar a desnormalização do banco de dados utilizado na aplicação móvel, por motivos previamente expostos;
- implementar um mecanismo de replicação/reconciliação híbrido, melhorando o nível de consistência.

Referências

AGRAWAL, P.; SREENAN, C. J. Get wireless: a mobile technology spectrum. *IT Professional*, v. 1, n. 4, p. 18–23, jul./ago. 1999. ISSN 1520-9202.

AHUJA, R.; BAGRODIA, R.; BAJAJ, L.; TAKAI, M. Evaluation of optimistic file replication in wireless multihop networks. In: *Global Telecommunications Conference, 1999. GLOBECOM '99*. Rio de Janeiro: [s.n.], 1999. v. 1, p. 259–265.

BARBARA, Daniel. Mobile computing and databases-a survey. *IEEE Transactions on Knowledge and Data Engineering*, v. 11, n. 1, p. 108–117, jan./fev. 1999. ISSN 1041-4347.

BARBARA, Daniel; IMIELINSKI, Tomasz. Sleepers and workaholics: Caching strategies in mobile environments. *VLDB Journal*, v. 4, n. 4, p. 567–602, 1995.

BARROSO, L. A.; DEAN, J.; HOLZLE, U. Web search for a planet: The google cluster architecture. *IEEE Micro*, v. 23, n. 2, p. 22–28, mar./abr. 2003. ISSN 0272-1732.

BAZGHANDI, A. Web database connectivity methods (using mysql) in windows platform. In: *Information and Communication Technologies, 2006. ICTTA '06. 2nd*. [S.l.: s.n.], 2006. v. 2, p. 3577–3581.

BEEK, J. J. van de; BORJESSON, P. O.; BOUCHERET, M. L.; LANDSTROM, D.; ARENAS, J. M.; ODLING, P.; OSTBERG, C.; WAHLQVIST, M.; WILSON, S. K. A time and frequency synchronization scheme for multiuser OFDM. *IEEE Journal on Selected Areas in Communications*, v. 17, n. 11, p. 1900–1914, nov. 1999. ISSN 0733-8716.

BERKENBROCK, C. D. M.; DANTAS, M. A. R. Investigation of cache coherence strategies in a mobile client/server environment. In: SUNDERAM, Vaidy S.; ALBADA, G. Dick van; SLOOT, Peter M. A.; DONGARRA, Jack (Ed.). *International Conference on Computational Science (3)*. [S.l.]: Springer, 2005. (Lecture Notes in Computer Science, v. 3516), p. 987–990. ISBN 3-540-26044-7.

BJORNSSON, Magnus E.; SHRIRA, Liuba. Buddycache: high-performance object storage for collaborative strong-consistency applications in a wan. In: *OOPSLA*. [S.l.: s.n.], 2002. p. 26–39.

CAI, Jun; TAN, Kian-Lee. Energy-efficient selective cache invalidation. *Wireless Networks*, v. 5, n. 6, p. 489–502, 1999.

CAI, Jun; TAN, Kian-Lee; OOI, Beng Chin. On incremental cache coherency schemes in mobile computing environments. In: *Data Engineering, 1997. Proceedings. 13th International Conference on*. Birmingham: [s.n.], 1997. p. 114–123.

CAO, Jiannong; ZHANG, Yang; CAO, Guohong; XIE, Li. Data consistency for cooperative caching in mobile environments. *Computer*, v. 40, n. 4, p. 60–66, abr. 2007. ISSN 0018-9162.

CHAN, Darin; RODDICK, John F. Context-sensitive mobile database summarisation. In: OUDSHOORN, Michael J. (Ed.). *ACSC*. [S.l.]: Australian Computer Society, 2003. (CRPIT, v. 16), p. 139–149. ISBN 0-909-92594-1.

CHEN, Hao. Update operation and partition dependency in distributed database systems. In: *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*. [S.l.: s.n.], 2003. p. 624–627.

CHEN, Lin; LENEUTRE, J. A secure and scalable time synchronization protocol in IEEE 802.11 ad hoc networks. In: *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*. [S.l.: s.n.], 2006.

CHUNG, H.; CHO, H. Data caching with incremental update propagation in mobile computing environments. *Australian Computer Journal*, v. 30, n. 2, p. 77–86, 1998. Disponível em: <citeseer.ist.psu.edu/chung98data.html>.

COATTA, T.; HUTCHINSON, N. C.; WARFIELD, A.; WON, J. H. T. A data synchronization service for ad hoc groups. *2004. WCNC. 2004 IEEE Wireless Communications and Networking Conference*, v. 1, p. 483–488, mar 2004. ISSN 1525-3511.

COGLIANESE, Michael. *Optimistic Data Replication for Mobile Applications*. 2000. Disponível em: <citeseer.ist.psu.edu/coglianese00optimistic.html>.

CORSON, M. Scott; FREEBERSYSER, James A.; SASTRY, Ambatipudi. Mobile ad hoc networking - editorial. *MONET*, v. 4, n. 3, p. 137–138, 1999.

CUNHA, D. P.; DANTAS, M. A. R. An experimental case study of replication on reconciliation in a wireless environment. In: ESKICIOGLU, M. Rasit (Ed.). *HPCS*. [S.l.]: University of Manitoba, Department of Computer Science, 2004. p. 179–182. ISBN 0-9735472-0-0.

DANTAS, Mario Antônio Ribeiro. *Tecnologias de Redes de Comunicação e Computadores*. Axcel Books, 2002. Hardcover. ISBN 85-7323-169-6. Disponível em: <http://www.axcel.com.br>.

EHSAN, H.; UZMI, Z. A. Performance comparison of ad hoc wireless network routing protocols. In: *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*. [S.l.: s.n.], 2004. p. 457–465.

FEINSTEIN, Wei Pan. A study of technologies for client/server applications. In: *ACM-SE 38: Proceedings of the 38th annual on Southeast regional conference*. New York, NY, USA: ACM Press, 2000. p. 184–193. ISBN 1-58113-250-6.

FERRO, E.; POTORTI, F. Bluetooth and wi-fi wireless protocols: a survey and a comparison. *IEEE [see also IEEE Personal Communications] Wireless Communications*, v. 12, n. 1, p. 12–26, feb 2005. ISSN 1536-1284.

FIFE, Leslie D.; GRUENWALD, Le. Research issues for data communication in mobile ad-hoc network database systems. *SIGMOD Record*, v. 32, n. 2, p. 42–47, 2003.

FISCHMEISTER, Sebastian; MENKHAUS, Guido; STUMPFL, Alexander. Location-detection strategies in pervasive computing environments. In: *PerCom*. [S.l.]: IEEE Computer Society, 2003. p. 273–278.

FLINN, J.; SATYANARAYANAN, M. Powerscope: a tool for profiling the energy usage of mobile applications. In: *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*. New Orleans, LA: [s.n.], 1999. p. 2–10.

FOX, G. Peer-to-peer networks. v. 3, n. 3, p. 75–77, maio/jun. 2001. ISSN 1521-9615.

FUKUSHIMA, T.; TAKAHASHI, E.; NARAZAKI, H. A wireless agent for automatic connection based on connection time prediction using rf information. *5th World Multi-Conference on Systemics, Cybernetics and Informatics*, p. 1–6, 2001.

GIACOMO, M. Di. MySQL: lessons learned on a digital library. *IEEE Software*, v. 22, n. 3, p. 10–13, maio/jun. 2005. ISSN 0740-7459.

GLITHO, R. H.; OLOUGOUNA, E.; PIERRE, S. Mobile agents and their use for information retrieval: a brief overview and an elaborate case study. *IEEE Network*, v. 16, n. 1, p. 34–41, jan./fev. 2002. ISSN 0890-8044.

GRAY, Jim; HELLAND, Pat; O'NEIL, Patrick E.; SHASHA, Dennis. The dangers of replication and a solution. In: JAGADISH, H. V.; MUMICK, Inderpal Singh (Ed.). *SIGMOD Conference*. [S.l.]: ACM Press, 1996. p. 173–182.

GUPTA, S. K. S.; SRIMANI, P. K. A strategy to manage cache consistency in a disconnected distributed environment. *IEEE Transactions on Parallel and Distributed Systems*, v. 12, n. 7, p. 686–700, jul 2001. ISSN 1045-9219.

HARA, Takahiro. Cooperative caching by mobile clients in push-based information systems. In: *CIKM*. [S.l.]: ACM, 2002. p. 186–193.

HELAL, A.; KHUSHRAJ, A.; ZHANG, J. Incremental hoarding and reintegration in mobile environments. In: *Applications and the Internet, 2002. (SAINT 2002). Proceedings. 2002 Symposium on*. Nara: [s.n.], 2002. p. 8–11.

HOLLIDAY, JoAnne; AGRAWAL, Divyakant; ABBADI, Amr El. Disconnection modes for mobile databases. *Wireless Networks*, v. 8, n. 4, p. 391–402, 2002.

HOU, Wen-Chi; SU, Meng; ZHANG, Hongyan; WANG, Hong. An optimal construction of invalidation reports for mobile databases. In: *CIKM*. [S.l.]: ACM, 2001. p. 458–465. ISBN 1-58113-436-3.

IMIELINSKI, Tomasz; BADRINATH, B. R. Data management for mobile computing. *SIGMOD Record*, v. 22, n. 1, p. 34–39, 1993.

ITANI, Z.; DIAB, H.; ARTAIL, H. Efficient pull based replication and synchronization for mobile databases. In: *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*. [S.l.: s.n.], 2005. p. 401–404.

ITANI, Z.; DIAB, H.; ARTAIL, H. Optimistic pull based replication for mobile databases. In: *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*. [S.l.: s.n.], 2005. v. 2, p. 895–900.

JIAO, Yu.; HURSON, A. R. Adaptive power management for mobile agent-based information retrieval. *2005. AINA 2005. 19th International Conference on Advanced Information Networking and Applications*, v. 1, p. 675–680, mar 2005. ISSN 1550-445X.

JIMENEZ-PERIS, Ricardo; PATINO-MARTINEZ, Marta; KEMME, Bettina; ALONSO, Gustavo. How to select a replication protocol according to scalability, availability, and communication overhead. In: *SRDS*. [S.l.]: IEEE Computer Society, 2001. p. 24–. ISBN 0-7695-1366-2.

KAHN, Joseph M.; KATZ, Randy H.; PISTER, Kristofer S. J. Next century challenges: Mobile networking for "smart dust". In: *MOBICOM*. [S.l.: s.n.], 1999. p. 271–278.

KIMM, Haklin; SHIN, S. Y.; SHIM, C. Y. Two approaches to improve java MIDP record management system in wireless devices. In: *Electro Information Technology, 2005 IEEE International Conference on*. [S.l.: s.n.], 2005.

KNAESEL, Frank Juergen. *SIGAF - Sistema Integrado de Gestão Acadêmica e Financeira*. fev. 2007.

KOTZ, David; GRAY, Robert S. Mobile agents and the future of the internet. *SIGOPS Oper. Syst. Rev.*, ACM Press, New York, NY, USA, v. 33, n. 3, p. 7–13, 1999. ISSN 0163-5980.

LAUZAC, Susan Weissman; CHRYSANTHIS, Panos K. Personalizing information gathering for mobile database clients. In: *SAC*. [S.l.]: ACM, 2002. p. 49–56.

LIM, J. B.; HURSON, A. R. Transaction processing in mobile, heterogeneous database systems. *IEEE Transactions on Knowledge and Data Engineering*, v. 14, n. 6, p. 1330–1346, nov./dez. 2002. ISSN 1041-4347.

LIU, George; MARLEVI, Alexander; MAGUIRE, Jr. Gerald Q. A mobile virtual-distributed system architecture for supporting wireless mobile computing and communications. In: *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1995. p. 111–118. ISBN 0-89791-814-2.

LIU, Jiangchuan; ZHANG, Qian; LI, Bo; ZHU, Wenwu; ZHANG, Jun. A unified framework for resource discovery and qos-aware provider selection in ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, ACM Press, New York, NY, USA, v. 6, n. 1, p. 13–21, 2002. ISSN 1559-1662.

LOH, Yin-Huei; HARA, Takahiro; TSUKAMOTO, Masahiko; NISHIO, Shojiro. A hybrid method for concurrent updates on disconnected databases in mobile computing environments. In: *SAC (2)*. [S.l.: s.n.], 2000. p. 563–565.

LORINCZ, J.; BEGUSIC, D. Physical layer analysis of emerging IEEE 802.11n WLAN standard. In: *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*. [S.l.: s.n.], 2006. v. 1.

MADRIA, S. K.; BHOWDRICK, S. S. Mobile data management. *IEEE Potentials*, v. 20, n. 4, p. 11–15, out./nov. 2001. ISSN 0278-6648.

MARIHART, D. J. Overview of mobile computing technology including 3g and 4g. In: . Vancouver, BC: [s.n.], 2001. v. 2.

MCDERMOTT-WELLS, P. What is bluetooth? *IEEE Potentials*, v. 23, n. 5, p. 33–35, dez./jan. 2004. ISSN 0278-6648.

MILLS, David L. Internet time synchronization: The network time protocol. In: *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*. [s.n.], 1994. Disponível em: <citeseer.ist.psu.edu/mills91internet.html>.

MOSTOFI, Y.; COX, D. C. Timing synchronization in high mobility OFDM systems. In: *Communications, 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 4, p. 2402–2406.

MUELLER, W.; SCHAEFER, R.; BLEUL, S. Interactive multimodal user interfaces for mobile devices. In: *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. [S.l.: s.n.], 2004.

NARAYANAN, Dushyanth; SATYANARAYANAN, Mahadev. Multi-fidelity algorithms for interactive mobile applications. *Wireless Networks*, v. 7, n. 6, p. 601–607, 2001.

OLSEN, C. M.; NARAYANASWARNI, C. Powernap: an efficient power management scheme for mobile devices. *IEEE Transactions on Mobile Computing*, v. 5, n. 7, p. 816–828, jul 2006. ISSN 1536-1233.

OZSU, Tamer M.; VALDURIEZ, Patrick. *Principles of Distributed Database Systems (2nd Edition)*. Prentice Hall, 1999. Hardcover. ISBN 0136597076. Disponível em: <http://www.amazon.co.uk/exec/obidos/ASIN/0136597076/citeulike-21>.

PARK, Taesoon; WOO, Namyoon; YEOM, Heon Y. An efficient optimistic message logging scheme for recoverable mobile computing systems. *IEEE Transactions on Mobile Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 1, n. 4, p. 265–277, 2002. ISSN 1536-1233.

PARK, Taesoon; YEOM, Heon Young. An asynchronous recovery scheme based on optimistic message logging for mobile computing systems. In: *International Conference on Distributed Computing Systems*. [s.n.], 2000. p. 436–443. Disponível em: <citeseer.ist.psu.edu/park00asynchronous.html>.

PATTEN, K.; PASSERINI, K. From personal area networks to ubiquitous computing: preparing for a paradigm shift in the workplace. In: *Wireless Telecommunications Symposium, 2005*. [S.l.: s.n.], 2005. p. 225–233.

PENG, Wen-Chih; CHEN, Ming-Syan. Design and performance studies of an adaptive cache retrieval scheme in a mobile computing environment. *IEEE Transactions on Mobile Computing*, v. 4, n. 1, p. 29–40, jan./fev. 2005. ISSN 1536-1233.

PHAM, Vu Anh; KARMOUCH, A. Mobile software agents: an overview. *IEEE Communications Magazine*, v. 36, n. 7, p. 26–37, jul 1998. ISSN 0163-6804.

PHATAK, Shirish Hemant; BADRINATH, B. R. Multiversion reconciliation for mobile databases. In: *ICDE*. [S.l.]: IEEE Computer Society, 1999. p. 582–589.

PITOURA, Evaggelia; BHARGAVA, Bharat. Building information systems for mobile environments. In: *CIKM '94: Proceedings of the third international conference on Information and knowledge management*. New York, NY, USA: ACM Press, 1994. p. 371–378. ISBN 0-89791-674-3.

PITOURA, E.; SAMARAS, G. *Data Management for Mobile Computing*. 1998. Disponível em: <citeseer.comp.nus.edu.sg/pitoura98data.html>.

PROMMAK, C.; KABARA, J.; TIPPER, D.; CHARNSRIPINYO, C. Next generation wireless LAN system design. In: *MILCOM 2002. Proceedings*. [S.l.: s.n.], 2002. v. 1, p. 473–477.

RATNER, David; POPEK, Gerald J.; REIHER, Peter. *The Ward Model: A replication architecture for mobile environments*. [S.l.], 1996. 30 p. Disponível em: <citeseer.ist.psu.edu/158431.html>.

RATNER, David; REIHER, Peter L.; POPEK, Gerald J. Roam: A scalable replication system for mobile computing. In: *DEXA Workshop*. [S.l.: s.n.], 1999. p. 96–104.

RATNER, David; REIHER, Peter L.; POPEK, Gerald J.; KUENNING, Geoffrey H. Replication requirements in mobile environments. *MONET*, v. 6, n. 6, p. 525–533, 2001.

SAITO, Yasushi; LEVY, Henry M. Optimistic replication for internet data services. In: HERLIHY, Maurice (Ed.). *DISC*. [S.l.]: Springer, 2000. (Lecture Notes in Computer Science, v. 1914), p. 297–314. ISBN 3-540-41143-7.

SAITO, Yasushi; SHAPIRO, Marc. Optimistic replication. *ACM Comput. Surv.*, ACM Press, New York, NY, USA, v. 37, n. 1, p. 42–81, 2005. ISSN 0360-0300.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. *IEEE [see also IEEE Wireless Communications] Personal Communications*, v. 8, n. 4, p. 10–17, aug 2001. ISSN 1070-9916.

SONG, Jin-Woo; PARK, Kyo-Sung; YANG, Sung-Bong. An effective cooperative cache replacement policy for mobile p2p environments. In: *Hybrid Information Technology, 2006. ICHIT'06. Vol 2. International Conference on*. Cheju Island, Korea: [s.n.], 2006. v. 2, p. 24–30.

TERRY, Douglas B.; PETERSEN, Karin; SPREITZER, Mike; THEIMER, Marvin. The case for non-transparent replication: Examples from bayou. *IEEE Data Eng. Bull.*, v. 21, n. 4, p. 12–20, 1998.

TRIFONOVA, A.; RONCHETTI, M. Mobile learning: Is anytime + anywhere = always online? In: *Advanced Learning Technologies, 2006. Sixth International Conference on*. [S.l.: s.n.], 2006. p. 702–706.

WIESMANN, M.; PEDONE, F.; SCHIPER, A.; KEMME, B.; ALONSO, G. Database replication techniques: a three parameter classification. In: *Reliable Distributed Systems, 2000. SRDS-2000. Proceedings The 19th IEEE Symposium on*. Nurnberg: [s.n.], 2000. p. 206–215.

WU, Kung-Lung; YU, P. S.; CHEN, Ming-Syan. Energy-efficient caching for wireless mobile computing. In: *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*. New Orleans, LA: [s.n.], 1996. p. 336–343.

WU, Shiow yang; CHANG, Yu-Tse. A user-centered approach to active replica management in mobile environments. *IEEE Transactions on Mobile Computing*, v. 5, n. 11, p. 1606–1619, nov 2006. ISSN 1536-1233.

XU, Bangnan; HISCHKE, S.; WALKE, B. The role of ad hoc networking in future wireless communications. In: *Communication Technology Proceedings, 2003. ICCT 2003. International Conference on*. [S.l.: s.n.], 2003. v. 2, p. 1353–1358.

XU, Jianliang; HU, Qinglong; LEE, Wang-Chien; LEE, Dik Lun. Performance evaluation of an optimal cache replacement policy for wireless data dissemination. *IEEE Trans. Knowl. Data Eng.*, v. 16, n. 1, p. 125–139, 2004.

XU, Jianliang; TANG, Xueyan; LEE, Dik Lun. Performance analysis of location-dependent cache invalidation schemes for mobile environments. *IEEE Trans. Knowl. Data Eng.*, v. 15, n. 2, p. 474–488, 2003.

YIN, Liangzhong; CAO, Guohong. Supporting cooperative caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, v. 5, n. 1, p. 77–89, jan. 2006. ISSN 1536-1233.

YUEN, Joe Chun-Hung; CHAN, Edward; LAM, Kam yiu; LEUNG, Hei-Wing. Cache invalidation scheme for mobile computing systems with real-time data. *SIGMOD Record*, v. 29, n. 4, p. 34–39, 2000.

ZACHARIADIS, Stefanos; CAPRA, Licia; MASCOLO, Cecilia; EMMERICH, Wolfgang. Xmiddle: information sharing middleware for a mobile environment. In: *ICSE*. [S.l.]: ACM, 2002. p. 712.

APÊNDICE A – Artigos

A.1 Publicados

.

BORGES, Vinícius da Cunha Martins; ROSSETO, Anubis Graciela Moraes; KNAESEL, Frank Juergen; DANTAS, Mário Antônio Ribeiro. An Enhanced Approach for Submission and Monitoring of Applications in the Mobile Grid. *5th International Symposium on Parallel and Distributed Processing and Applications - ISPA 2007*. Niagara Falls, ON, Canada. Aug/2007.

.

KNAESEL, Frank Juergen e DANTAS, Mário Antônio Ribeiro. Sharing Consistent Data in Ad-Hoc Networks. *V CONGED - Congresso de Tecnologias para Gestão de Dados e Metadados do Cone Sul*. Cascavel, PR, Brasil. Aug/2007.